DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A GUIDE TO MACRO AND MICRO COMPUTER PERFORMANCE EVALUATION

by

Gary Kenneth Gray

December 1983

Thesis Advisor: Alan A. Ross

Approved for public release; distribution unlimited

T215182

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): A Guide to Macro and Micro Computer Performance Evaluation	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis, December 1983
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Gary Kenneth Gray	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943	12. REPORT DATE: December 1983
	13. NUMBER OF PAGES: 124
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer performance evaluation, computer performance management programmed tuning

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Guidelines and discussions are presented for computer performance evaluation at two levels. The first level, Computer Performance Management (CPM) or Macro Performance Evaluation, involves an overall computer performance management strategy concerning the use of computer resources. The role of CPM throughout the computer system life-cycle is also discussed. The second level of computer performance involves (Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT (Continued)

Computer Performance Evaluation (CPE) or Micro Performance Evaluation. A brief discussion of CPE tools is given, as well as how to select a performance monitor. Some computer performance fallacies are revealed and a discussion of the determination of "critical sections" of software systems and program tuning practices for improving system performance is presented. Limited discussion is devoted to performance issues in relatively new areas in the computer field such as networks, data base management systems and microcomputers.

Approved for public release; distribution unlimited.

A Guide to Macro and Micro Computer Performance Evaluation

by

Gary Kenneth Gray
B.S., Saint Mary's College of Maryland, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1983

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943

ABSTRACT

Guidelines and discussions are presented for computer performance evaluation at two levels. The first level, Computer Performance Management (CPM) or Macro Performance Evaluation, involves an overall computer performance management strategy concerning the use of computer resources. The role of CPM throughout the computer system life-cycle is also discussed.

The second level of computer perfomance involves Computer Performance Evaluation (CPE) or Micro Performance Evaluation. A brief discussion of CPE tools is given, as well as how to select a performance monitor. Some computer performance fallacies are revealed and a discussion of the determination of "critical sections" of software systems and program tuning practices for improving system performance is presented.

Limited discussion is devoted to performance issues in relatively new areas in the computer field such as networks, data base management systems and microcomputers.

TABLE OF CONTENTS

I. OBJECTIVE 11

II. INTRODUCTION 15

III. COMPUTER PERFORMANCE MANAGEMENT 21

A. GETTING STARTED--THE FIRST STEP 22

1. Selecting CPM Team Members 23

2. Training The CPM Team 27

B. CPM RESPONSIBILITIES 28

1. User Requirements 29

2. User Support 34

3. User Resource Utilization Report 35

C. RESOURCE MANAGEMENT 36

1. Communicating With Upper Management . . . 36

2. Vendor Relations 37

D. INSTITUTING THE CPM PROGRAM 38

1. CPM Reporting 39

2. Management's Role 41

3. Resource Requirements 47

IV. CPE TOOLS AND THE SYSTEM LIFE CYCLE 48

A. CPE TOOLS 48

1. Accounting Data Reduction Programs 48

2. Software Monitors 51

3. Program Optimizers 52

4. Hardware Monitors 53

5. Benchmarks 54

6. Modeling 54

B. CPM AND COMPUTER SYSTEM LIFE CYCLE 55

1. Procurement Phase 55

2. Installation Phase 57
3. Operations Phase 58
4. Transition Phase 59
C. PRACTICAL ARCHITECTUAL ENHANCEMENTS 60

V. CHOOSING A MONITOR-HARDWARE OR SOFTWARE? 65
A. SOME GENERAL CONSIDERATIONS 65
1. Characteristics of Hardware and Software Monitors 66
B. MAKING THE DECISION 72

VI. SOME COMMON SENSE PERFORMANCE FALLACIES 73
A. SOME SPECIFIC FALLACIES 73
B. SOME POSSIBLE FALLACIES 79
C. GENERAL FALLACIES 81
D. A SYSTEM PERFORMANCE IMPROVEMENT PROCEDURE . . 82
E. SOME POSITIVE GUIDELINES 86

VII. IMPROVING PERFORMANCE WITHOUT MONITORS 88
A. AN APPROACH TO FINDING CRITICAL PARTS 90
1. Check Out the Compiler 90
2. Efficiency Measurement 92
3. Obtaining the Efficiency Units 93
4. Finding Critical Job Steps 94
5. Critical Job Step Selection 94
6. Tune the Program and Compare the Results 94
B. PROGRAM TUNING HINTS 95

VIII. CURRENT AND FUTURE TECHNOLOGY AND PERFORMANCE . 100
A. END-USER COMPUTER POWER 100
B. MICROCOMPUTER PERFORMANCE 100
1. Communication 101
2. Compatability 101
3. User Interface 102

4. Inexpensive Software 102
C. NETWORKS 102
1. Communication Line Varification 106
2. Data Verification 106
3. General Performance Summary 107
4. Performance Checking Intel Network . . . 109
D. DATA BASE MANAGEMENT SYSTEM (DBMS) PERFORMANCE 110
1. Model Application 111
2. Model DBMS Performance Related Outputs . 112

IX. CONCLUSION 114

APPENDIX A: SOURCES OF INFORMATION 117

APPENDIX B: EXAMPLES OF PERFORMANCE MEASURES 119

LIST OF REFERENCES 122

INITIAL DISTRIBUTION LIST 124

LIST OF TABLES

I. CPE Tools and System Life Cycle 61
II. Characteristics of Hardware and Software Monitors 67
III. Response Time Components 75

LIST OF FIGURES

2.1	A Common Approach to Performance Efforts	17
3.1	The Computer System Life Cycle	40
3.2	Turnaround Time Report	42
3.3	System Reliability Report	43
3.4	User Support Report	44
6.1	Recommended Performance Improvement Procedure	83
6.2	Suggested Hypothesis Testing Procedure	84
8.1	Data Communications Testing Environment	105
8.2	Trouble-Shooting Algorithm	108

ACKNOWLEDGEMENTS

The author wishes to thank the following people. Mr. Roger Martin and his staff at the Naval Post Graduate School Dudley Knox Library for their attenative and most efficient help in securing reference material almost overnight. Dr. Alan Ross, my adviscr, for his guidence and meticulous screening of what I ment to say and his suggestions on how to more correctly and understandably present it. Captain Brad Mercer, my seccnd reader, for making time in his busy schedule to provide additional help, insites and assistance in completing this thesis. Finally, and most impcrtantly, my wife, Ruth and my childern Corey, Scotty and Timmy for their patience, understanding, moral support, comfort and liberal leave policy throughout this effort.

I. OBJECTIVE

A computer system is a complex and dynamic mixture of hardware, software and human resources that interact together to provide a service. Early computers had slow, expensive and limited resources. The need to increase the performance of the system to best take advantage of the these expensive limited resources became a challenge and a necessity. On a batch system performance evaluation was far less complicated then on the multi-task operating environments of the third generation computers. Computer Performance Evaluation (CPE) of a computer system can easily become very expensive in many areas: system resources, manpower, and financial. Even after large expenditures in all of these areas, the results are usually hard to interperet or understand and may lead to meaningless conclusions that can actually be of little value in providing improved performance.

In order to achieve successful and meaningful CPE it is necessary to use Macro Computer Performance Evaluation, which basically means to design, develop and implement a plan or framework to guide and direct CPE efforts. This involves the coordination of both a Computer Performance Management (CPM) strategy and a technical evaluation (measurement effort) strategy. Obviously, a CPM strategy without a technical evaluation strategy for gathering measurements will not produce any useful results. On the other hand, a technical evaluation or Micro Computer Performance Evaluation, strategy without a well thought out CPM will most likely lead to the situation of questionable or useless results mentioned earlier.

The objective of this thesis is to provide a guide for building the foundation for both the CPM component and the CPE technical evaluation component for a computer performance program. The effective use of computer resources will be emphasized and some technical details and expertise required to carry out the measurement and analysis phase of CPE will be introduced. Emphasis will be placed on the idea that it is not necessary to get involved in the more sophicated and costly CPE tools such as software monitors and hardware monitors in order to obtain noticeable performance enhancements.

It is acknowledged that guides for CPE are not new and much research has been completed producing abundent information. However, the general approach of past research tends to view performance issues from the internal component perspective of the computer system and the emphasis is on the efficient use of limited, expensive hardware and software resources. This traditional view caused most CPE efforts to rely upon expensive and sophisticated hardware and software monitoring tools. Today, hardware is getting less and less expensive and is therefore no longer considered a critical limited resource. More and more computer power is given to the end user and the emphasis of CPE is switching from the efficient use of the system to effective utilization of the system. This guide will place more emphasis on the effective use of the system instead of the traditionaly quantatively oriented, efficient use of the computer system. The hardware maybe working very efficiently at transferring data from secondary storage to primary storage, but the user may not have blocked the records. Much more effective use can be made of the computer resources by blocking the file, therefore increasing the performance of the computer.

The traditional concept of computer performance evaluation must broaden it's scope from a microscopic internal resource concentration to a more macroscopic external resource, total system evaluation CPM/CPE effort. People and other indirectly coupled resources can have a significant impact on the performance of the system. Many computer facilities require tremendous amounts of an organization's resources to implement and operate. Performance of these computer systems must be managed to avoid critical performance shortcomings that jeopardize the needs of the organization. Traditional CPE tools and methods as well as new ones will support the CPM effort.

The guide is intended for both managers and technical personnel who are concerned about computer performance and/or are about to become involved in a CPE endeavor. Managers will be able to obtain a feel for the scope and depth of effort as well as the support required to perform CPE effectively. Inexperienced technical people will be able to find out what tools and methods are avaliable for performing measurements.

A second objective of this thesis is to explore the impact new trends and endeavors in the field will place on the current concept of CPE. It appears that current CPE capability will not be adequate for networks (individual systems connected by communication links), array processors (systems containing thousands of processors), or Data Base Management Systems (DBMS). Ideas in some of these areas are presented in Chapter VIII "Current and Future Technology and Performance".

Appendix A, "Sources of Information", is intended to provide both managers and technical personnel with additional sources of information in their areas of interest. Appendix B, "Examples of Performance Measures" provides a listing of some of the more common performance measurements and arranges the measurements into five groups:

1. Quantity of work performed
2. Quality of service
3. Component utilization
4. Underlying factors
5. Workload characteristics

II. INTRODUCTION

"The first computer had no more than two operational programs before the idea occured to insert check flags into the coding to obtain a measure of how far the program had gotten before it, or the machine, ceased operation" [Ref. 1]. Surprisingly, most manufacturers of computer systems as well as managers and users of these systems have not accomodated this interest in performance to any appreciable degree. According to Olson [Ref. 2] "Data processing activities have been reluctant in the past to address firm performance requirements." This reluctance on the part of all personnel associated with the computer may be attributed in part to the feeling that if no set standard exists, they can never be held accountable for not meeting it. The preliminary investment in time and resources required by most computer performance efforts can also be a major factor in postponing a serious interest in performance until a crisis erupts demanding immediate performance attention. Some simple architectural modifications could significantly aid the hardware monitoring effort; this is discussed further in the section on hardware monitors. One of the oldest indicators of performance is the CPU busy (or CPU wait) light which is usually located on the front console of the computer. However, the CPU busy light was not intended for performance observation, but rather for maintenance applications and it is hard to measure the number of microseconds of idleness by examing the CPU busy light.

Morris also points out that "the two most important and effective tools for evaluation of computer performance are available free in nearly every computer installation. They are visual inspection and common sense". For example, if

the busy light is constantly on (meaning CPU is busy) while the program is executing, then the program could be CPU bound. Perhaps program tuning could be used to improve the performance of the program. If the CPU light is constantly off and the disk units are "dancing about" the computer room floor, this indicates that the programs executing are I/O intensive, and perhaps the paging algorithm could be adjusted or the blocking buffer size should be changed. Blocking buffer size is an important performance issue and is discussed in detail in a later section. This could also indicate that there may be very little "overlap" achieved between the CPU and I/O. Some fallacies that can result from the use of these tools are discussed in Chapter IV "CPE Tools and the System Life Cycle.

Another example, presented at a computer performance conference, concerns the degraded performance of a system between the hours of 11 A.M and 1 P.M. Shortly after the installation of the new version of the operating system, the System Administrator (SA) began receiving compaints about the increased turnarcund time for jobs for the period corresponding to the two hour period in which most people took a lunch break. The SA was very puzzled since prior to the installation of the revised operating system, turnaround time on the old operating system usually improved during this period.

After several days of inquiring about new jobs run, and having uncovered no new results, the SA called the vendor's system analyst out of frustration. The SA mentioned the possible need for a software or hardware monitor to the analyst. The analyst suggested that they first look at a dump of the accounting data for a period shortly before and after the installation of the revised operation system. Several new "system programs" began to dominate the job queue shortly after the installation of the revised version

of the operating system and during the 2 hour lunch period. Upon looking at the source file of these programs, the analyst discovered that the majority of them were from a new "package" included in the revised version of the operating system compliments of the vendor. This package turned out to be a collection of games used by the vendor for promotional purposes. The employees of the computer installation had renamed the games using system-like names, to avoid being easily detected, and were playing the games during lunch time.

This example demonstrates the vendor systems analyst's wise choice of using the two most important CPE tools: visual inspection and common sense. Common sense should also prevent any further, more detailed investigation beyond this first casual "visual inspection" without taking the time to develop a sound computer performance management plan. To proceed without such a plan would be expensive, wasteful and most likely produce unsatisfactory, possibly misleading results. A common sense initial approach would use the 80-20 rule, i.e. 80% preparation, and 20% perspiration. Regrettably, the most common approach is depicted in figure 2.1 obtained from Bell [Ref. 3].

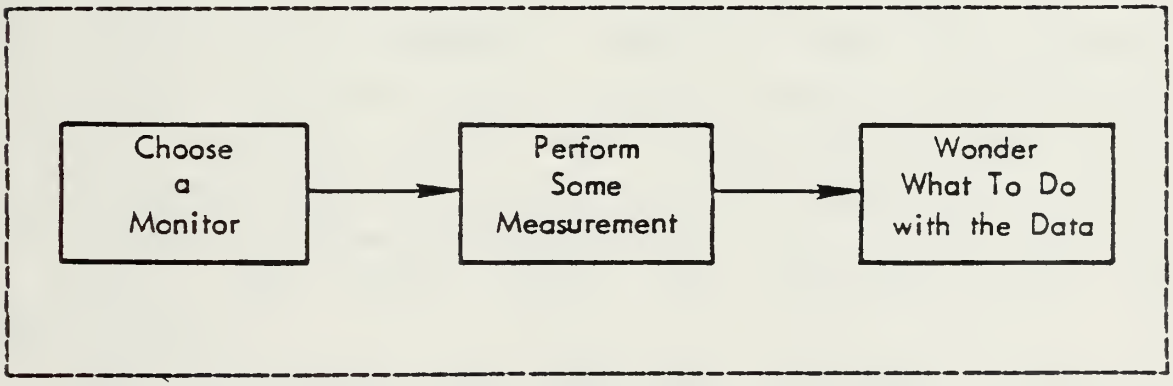


Figure 2.1 A Common Approach to Performance Efforts.

CPE is usually first considered after a system has been used for a period of time and becomes unable to provide satisfactory performance to either a specific program or to the whole system in general. Actually, CPE could be performed at all phases of the life cycle of a computer system, which typically includes the following phases:

- Procurement Phase
- Installation Phase
- Operation Phase
- Transition Phase (to a new system)

Due to economic reasons, formal life cycle CPE is usually restricted to large mainframes and super computers supported by large enough budgets to afford this effort. Minicomputer systems usually receive inexpensive, very informal CPE in the first two phases where the predominate goal is to buy as much hardware (main memory and secondary storage-disk capacity) as their procurement budget will support from the vendor who has the best reputation for quality system software and software development tools. This approach is most likely motivated by the fact that the most expensive item in a computer system's long term budget is software development costs, not hardware costs. However, the software is extremely dependent upon the hardware purchased, and relies upon a certain level of performance from this hardware. If the issue of hardware performance is not adequately addressed during the procurement phase, all the expensive software developed will be of little value on a system that can't supply the required level of performance.

Once the decision has been made to attempt CPE on a system, a CPM program should be initiated, and it's first

task should be to establish a strategy or define a set of CPE objectives and their scopes. This is covered in more detail in Chapter III "Computer Performance Management".

When attempting to locate potential areas and bottlenecks affecting system performance the two most important and effective tools for evaluation: visual inspection and common sense should be used. Most of today's systems have accounting packages already installed and are collecting valuable CPE information, mainly about the workload. A visual inspection of the accounting data will give a clue to what kinds and types of jobs use the most resources on the system. Use of software monitors and hardware monitors should be put off until they are absolutely needed and can be proved to be cost effective. Once a hardware monitor is introduced into the CPE effort the expertise and technical resources required to support the hardware monitor in order to gain meaningful and beneficial results will most certainly become one of the most expensive items in the CPE budget. Chapter IV CPE Tools and the System Life Cycle" and Chapter V "Choosing a Monitor--Hardware or Software?" provides information in this area.

Personal computers (PC's) and networks are evolving and gathering respect for their ability to solve a wide range of problems. This is an indication that decentralized technology and end user computer power is on the rise. This means that people no longer have to go to the computer in the sense that they must leave their office to obtain the services of the computer. Advances in computer communication technology has made it possible to take the services of the computer to the users in their office. Decreasing costs as a result of advances in integrated circuit technology has made it possible to provide local computer capability at reasonable costs. Performance of the "system" then, concerns global as well local considerations.

Our traditional views of CPE, as well as our methods, are being challenged by the emergence of new and expanding computer technology which demands new methods and approaches to perform meaningful CPE. On the other hand, there are easy to use performance improvements, presented in Chapter VII "Improving Performance Without Monitors", that do not receive wide spread use, but could significantly improve the performance of a system. Chapter VI "Some Common Sense Performance Fallacies" reinforces the idea that hypotheses about performance issues must always be validated by measurements.

III. COMPUTER PERFORMANCE MANAGEMENT

Before any professional carpenter reaches for his measuring tape to cut a peice of lumber, he will always consult the 'blue print' or plan first. This is usually a mental exercise in most cases, for he has previously gone over the actual plan many times in great detail, but he also returns to it frequently to review critical measurements. The important issue is that he works from a plan. This saves time, coordinates his activities toward a predetermined goal and avoids waste. The above approach should also be relevant for anyone involved in CPE.

In FIPS PUB 49 [Ref. 4], a CPM program is defined as "any structured effort, in house or otherwise, to measure and evaluate the performance of a computer facility in support of established management goals and objectives". In addition, the CPM program should strive to maintain the computer system at the highest performance-to-cost ratio and provide management with reports on the status of the computer system. These reports should be simple, short, and easily understandable by management.

The immediate objective of a CPM strategy is not to troubleshoot the system or to do system tuning, but rather to maintain, or in some instances, regain control of a complex, costly and critical resource through a quantitative and qualitative understanding of how that resource performs and of the alternatives that are avaliable to make it perform more effectively and efficiently. Thus, CPM is obligated to serve management, who initially approved funding for the computer facility resource, and to the users who must use the system to meet the organization's established goals and objectives.

A. GETTING STARTED--THE FIRST STEP

Management must realize that computers directly affect an organization's bottom line. This effect is expected to increase substantially in the future, due to the unusual nature of the computer products market where physical size of the system is decreasing, capacity and capability are increasing and prices are decreasing. Thus, the mainframe computer of yesterday, usually found in a computer room away from people, is now showing up in small packages on the tops of desks in the office enviornment.

Spinelli [Ref. 5] defines the role and responsibilities of the information-systems manager who will most likely rely upon the CPM group to provide most of the information he needs. Therefore, the CPM manager must be aware of this role and be capable of supporting the following responsibilities:

1. Enable the organization to understand, apply and control internal forces, and properly utilize external forces, that shape a firm's computing environment.
2. Apply proven management principles, particularly strategic planning, to information-systems functions.
3. Provide the organization with planning and development concepts and tools that effectively enable the firm to develop and implement a corporate computing/business strategy, and to provide a logical framework in which it can understand newest usage trends in information systems management and computer technology.
4. Provide the organization with a forum to exchange ideas and discuss opportunities with information systems specialists.

5. Comprehend the critical nature of information systems functions in the context of overall organizational success.

A wide range of computer performance measurments are the primary source of information for the above areas.

In the aggregate, computer systems represent the totality of the organization itself, in that, one vital corporate resource--information--is provided to the other vital corporate resource--people. Therefore, timely relevant, accurate information, effectively produced and disseminated, is a vital corporate resource. In this light, CPM must use CPE in providing this resource in the most efficient and effective manner.

Regrettably, one external reason for the existence of a CPM group in the Federal Government and the DoD is the extremely long procurement time required to order new equipment. Typically, if the workload on a system becomes too much and extensive enhancements or a new system is needed, it will take a minimum of one year to obtain the desired equipment using the extremely long and paperwork intensive ADP procurement process required by the Federal and DoD.

On the positive side, steps must be taken to maintain the best system performance during this procurement time, which motivates the organization to implement CPM and CPE.

1. Selecting CPM Team Members

In order to develop a CPM plan it is necessary to select personnel to form a CPM team or group. It's politically wise to select at least one represenative from each of the various departments that can directly influence the performance of the computer system in order to insure overall cooperation. It is both politically wise and economically essential to include a representative from

upper management, since they traditionally are responsible for allocating funds for such groups and can give the CPM group the needed recognition and importance to sustain an effective role in the organization.

Each representative need not be a full time active member of the group, but should remain aware of the main goal of the CPM effort. Full time participation of all members is not very practical in a large organization with many departments, since it would tend to make the CPM group too large and possibly ineffective. Small organizations could have difficulty also, since it may not be possible to dedicate many people to this effort on a full time basis due to lack of personnel. If possible, a minimum 'core' of fulltime CPM personnel should be choosen, that are well insulated from "fire fighting" and the day to day disjoint crisis situations which usually exist in a computer system work environment to insure continuity in the CPM effort.

If the organization forming a CPM group is part of the Federal Government or the Department of Defense (DoD), two organizations can provide a great deal of help since both organizations are dedicated to helping in the area of CPM and CPE. The Federal Government organization is known as FEDSIM (Federal Automatic Data Processing Simulation Center). See [Ref. 6] for location and contact information. Quoting from the organization's brochure: "FEDSIM provides a central source for computer simulation services so that individual agencies will not need to develop independent capabilities to use advance techniques of computer performance measurement and evaluation. This will allow all agencies to have access to powerful techniques on a cost reimbursement basis without incurring high individual start-up costs in time, money and expertise".

FEDSIM policies are established by a Joint Policy Committee made up of representatives of GSA, Air Force, the

Office of the Secretary of Defense and the National Bureau of Standards. FEDSIM provides the following services, tools and techniques:

1. SERVICES
   a) Computer performance evaluation consultant services and technical assistance for:
      i) Systems design
      ii) Systems specification
      iii) Computer equipment configuration
      iv) Computer program improvement
      v) ADP equipment selection
   b) Contractual assistance for purchase, lease or use of:
      i) Simulation packages and languages
      ii) Hardware monitors
      iii) Software monitors
      iv) Analytical techniques
   c) Training in the application of computer performance measurement and evaluation techniques
2. TOOLS AND TECHNIQUES
   a) Simulation Packages
      i) COMET, Computer Operated Machine Evaluation Technique (commercially available as SCERT, Systems and Computer Evaluation and Review Technique)
      ii) CASE, Computer Assisted System Evaluation
      iii) SAM, Systems Analysis Machine
   b) Simulation Languages
      i) GPSS, General Purpose System Simulation
      ii) SIMSCRIPT
      iii) GASP, General Activity Simulation Program
   c) Hardware Monitors
   d) Software Monitors

e) Analytical Routines

FEDSIM was established by the GSA and is operated by the Air Force at the request of GSA.

The second organization is the Navy Regional Data Automation Center (NARDAC), Pensacola. See [Ref. 7] for the complete organization title and location. This organization has the responsibility to provide tools, methods, and procedures for Computer Performance Evaluation/Capacity Management, as well as the resposibility to provide training and consultation services to Navy activities as required.

According to S. B. Olson [Ref. 2] "The first fact to come to light was that research indicated that effective Performance Management functions were simply not being accomplished within Navy data processing activities. In fact, evidence suggested that throughout industry as well, there was relatively little well organized and systematic Performance Evaluation or Capacity Planning being done."

One of the primary reasons perceived by Olson as hampering good performance and capacity management was the "general lack of a stand-alone specialized group whose primary responsibility was the performance management effort". Typically, it was found, when studies were made or data collected, the effort was usually assigned, perhaps by default, to a person or group of day-to-day production people at the data processing center.

Obviously, as pointed out earlier, in a small group, this problem would be hard to eliminate without a noticable loss of support in other areas. However, it does indicate that for larger groups, management was either not involved or involved but unaware of the potential benefits of a well organized CPM program and therefore treated the CPM efforts with low or no priority when "fires" occured. One possible solution to this problem is to educate management as to the potential benefits of a CPM program and assign a priority to

c. Reliability

Frequent system crashes or down time can be a nuisance, especially to the interactive users. Information should be recorded on mean-time-between-failure (MTBTF) and mean-time-to-repair (MTTR). A high frequency of system failures could indicate a weak hardware or software link or bug in the system. Problems in this area are very hard to solve due to inadequate information. Again, accounting log files could help uncover a pattern, possibly related to a particular job or combination of jobs running concurrently that could be related to the system problem. More discussion on this subject is in the section on Vendor relations.

d. Availability

In a very broad sense availability is the percent of the total time the system can directly serve the users to help meet the goals and objectives of the organization. Scheduling, system maintenance, system failures and repairs influence system availability. An extreme example dealing with both availability and accessibility concerns the solution of one computer facility to a problem of insufficient terminal availability. There were simply not enough terminals for the demand of users who needed them. The "solution" devised by the system administrator was to allow users to log on only once each day. Any attempt to log on the system again, in the same day by the same user, would be denied. This "solution" caused people to "hang on" to their terminal by running endless jobs while they were not actually using the terminal, so that they could have it available, if needed later in the day. A better solution may have been to extend the shift and schedule users for specific terminals.

2. User Support

Support functions such as training seminars, user consultation, system documentation and the need for regular computer center news bulletins are difficult requirements to determine and evaluate.

Once again, the availability of detailed accounting data can be of use in this area. Frequency or infrequency of use by users of system software tools and system resources in general, as well as frequency of compiles can help determine which users may require initial training or additional training in the use of system resources. A new programmer with a very optimistic outlook always included the catalog job control statement after the compile statement in all software under development. Consequently, the operating system would catalog a job even though the compiler encoutered fatal errors and the compile was unsuccessful. The programmer was needlessly wasting system resources and contributing to inefficient use of the computer. Apparently, the programmer was unaware that the operating system did provide for a conditional job control statement which would have prevented the catalog if the compile were unsuccessful. Or, possibly the programmer added or deleted some code to an existing program and failed to take out the catalog statement until assured that the compile was successful. If a new software tool is infrequently used, a training session may be required.

It is also important to maintain a well run and responsive user problem report or suggestion/comment system. Users are a very good source (sometimes the only source) for finding system hardware and software problems or bugs. Users can be encouraged to participate in the system by always providing quick and immediate sincere feedback, if only to reveal the status of their report or comment, and thank them.

3. User Resource Utilization Report

Many computer facilities must use accounting data for user charge back systems which are necessary to obtain the funds required to operate and maintain the computer facility. These reports are usually very high level, generally providing totals for categories of resource usage, but the system uses more detailed information to generate these totals. This detailed information about user resource utilization should be made into a User Resource Utilization Report and given to each user showing by job, what resource is being used and at what amount. The user should be adequately trained to take advantage of this information. Once accountability for computer resource utilization is established and made known to the users, they are frequently encouraged to cut down on wasteful use of computer resources. This can be especially true in the case when one user who needs a resource or more of a resource, knows what user or users are possibly wasting that resource or using it inefficiently. The net effect can be a reduction of user demand for critical resources and an increase in system performance.

It should be mentioned that accounting packages span the range of levels of detailed information they collect about jobs in the system and about the system. Traditionally, larger, more expensive systems, like IBM, have accounting systems whose log files contained more detailed information about individual jobs (number of I/O requests, amount of memory used, number of pages of printed output, etc.) and the system; whereas smaller, less expensive systems have accounting data of less detail. In few cases is it found that the accounting log files contain all the information required, and supplemental software is required to gather this additional information. Reviewing

the source code listing and the documentation of an accounting program can proivide a great deal of information about what, where and how to gather data on job and system performance.

C. RESOURCE MANAGEMENT

The CPM group should be able to easily monitor which resources are underutilized (for example cardreaders, 7 track tape units, etc.) and which resources are having difficulity meeting the workload demands. Using this information, obtainable through the accounting package and modifications to it, the CPM group will be able to make a report, reguarding elimination of under utilized resources (shifts, equipment, etc.) or the procurement of additional resources (memory, disk drives, tape drives, etc.). Through these computer performance evaluation efforts, direct cost savings as well as improved performance of the system can be realized by the computer facility.

Information collected in a historical database in this area can be an invaluable aid in predicting the near and long term pattern of growth expected for the computer facility. "Number of jobs completed per month, percent utilization of major system resources, hours of system availability are several measures applicable to this problem. Performance data is therefore valuable not only for enhancing the present system, but also for constructing models of future resource requirements". [Ref. 4]

1. Communicating With Upper Management

It should be a prime responsibility of the computer facility to report to upper management on the status of how well the computer facility is meeting it's part in supporting the organizations objectives and goals. Computer

system performance is a key issue in this commitment and the CPM group can provide much of the required information and recomendations in this area. Reports should be utilized to convey this information to both the computer facility and to upper management. In addition to these reports the computer facility manager may require additional information about the performance of the system at a more detailed level.

The form and format of these reports can vary greatly due to each organizations reporting requirements, but some general guidance can be given [Ref. 1], [Ref. 4] :

- Status reports should be regular, concise, and preferably graphical in nature.
- The amount of information reported should not exceed management requirements. "Too much, too often" is a problem common to many performance reporting schemes.
- Information should be at a level of abstraction which upper management can easily digest and understand, but sufficient to support the decision making process.
- The reports should compare the center's current level of performance against a set of predefined goals.

2. Vendor Relations

Most computer facilities interface regularly with vendor marketing and technical support personnel. A positive relationship with technical personnel can be invaluable especially in the areas of hardware and software reliability and maintenance. A joint review of hardware and software performance data on a regular scheduled basis by the computer facility personnel and vendor representatives will help foster mutual respect and understanding. Tracking system problems such as the frequency of tape and disk

parity errors, the cause and duration of system crashes, and variations in other system performance measures will provide facility manager with a folder of factual information to document the occurrence of these problems with which the vendor must deal. The capability to identify the source as well as the existence of errors becomes especially important in the multi-vendor computer system enviornment, in which each vendor points at the other vendor as being the main source of the problem.

Any proposed hardware and/or software modification will most certainly affect the performance of the system. They must therefore be evaluated from the total system point of view, not from just one aspect, as to:

- Utility
- Cost Effectiveness
- Impact on total System Performance

"A modification to a system scheduler which is intended to increase batch throughput but which inadvertently degrades interactive response time fails to consider total system performance. Measurement tools and techniques can be used not only to detect performance problems, but also to anticipate them and to prevent their occurrence." [Ref. 4]

D. INSTITUTING THE CPM PROGRAM

Measurement and evaluation techniques are available to support the efficient and effective management of a computer facility. How to introduce them into the computer system enviornment can be a problem. Some questions from [Ref. 4], properly asked are:

- How often should the information obtained from performance data be reported to the computer facility administrator?

- In what form should it be reported?

- What is the role of the computer facility administrator in instituting CPM procedures?

1. CPM Reporting

Figure 3.1 [Ref. 4], shows typical computer system life-cycle, progressing from an analysis of requirements to the final installation, operation, and enhancement of the selected system. In each phase of the computer life-cycle performance measurement and evaluation should provide a base for satisfying the informational needs of the computer facility administrator. Performance data is as useful during the requirements analysis phase as it is during the system enhancement phase. Every installation, reguardless of size, should incorporate a system performance reporting strategy during each phase of its system's life-cycle.

Availability alone should not be used to determine the types of data to be collected and reported. Instead, the data should first be selected as to the informational requirements of the computer facility, which are determined by the computer facility administrator's scope of responsibility. Each report should provide a historical trend of the facility's performance. Updates should be performed on a regular basis (depending upon the nature and importance of the information), and should contain specified performance criteria. Control limits should be established from this criteria and placed on the performance charts. Control limits are a values chosen to represent the boundaries of acceptable performance for a given system variables. These variables and their associated control limits could be

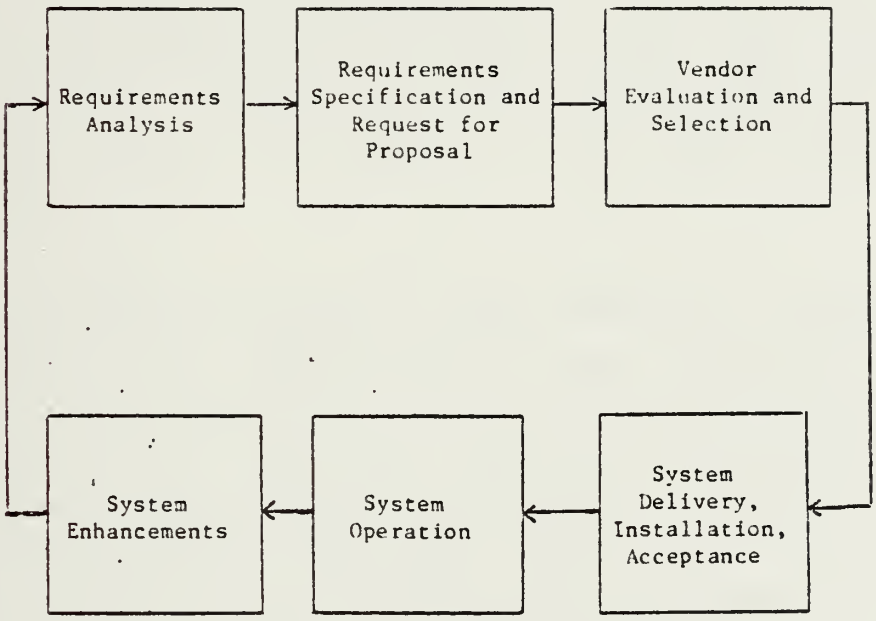


Figure 3.1 The Computer System Life Cycle.

objective-directed and indicate the facility's ability to meet certain specified objectives such as one-hour average batch turnaround time and 3 second terminal response time. Others are process-directed indicating the level of performance of internal system resources like the CPU, disks, and memory. Exception reports should be generated whenever control limits are exceeded. When appropriate, an in-depth study may be recommended to determine the specific cause(s) for the exception and appropriate solution for its correction. Reports should be simple and consistent with the computer facility's responsibility to it's users in the areas of turnaround time, reliability, and user support; see figure 3.2 for a turnaround time report, figure 3.3 for a reliability report, and figure 3.4 for user support report [Ref. 3].

Determining control limit values is dependent upon configuration and capabilities of each individual computer facility, as well as on the goals of the organization. Past performance may be used with caution as a standard against which current performance is evaluated. Past performance does not necessarily indicate good or poor performance, although it is a reliable indicator of the baseline system's natural reaction to various workload demands.

The perfomance reports generated by the CPM group should always be made available to and discussed with the user community. Publication of "performance charts" in the facility's newsletter or on the bulletin board is an excellent vehicle for accomplishing this.

2. Management's Role

The computer facility administrator (management) should play a major role in the implementation and long term

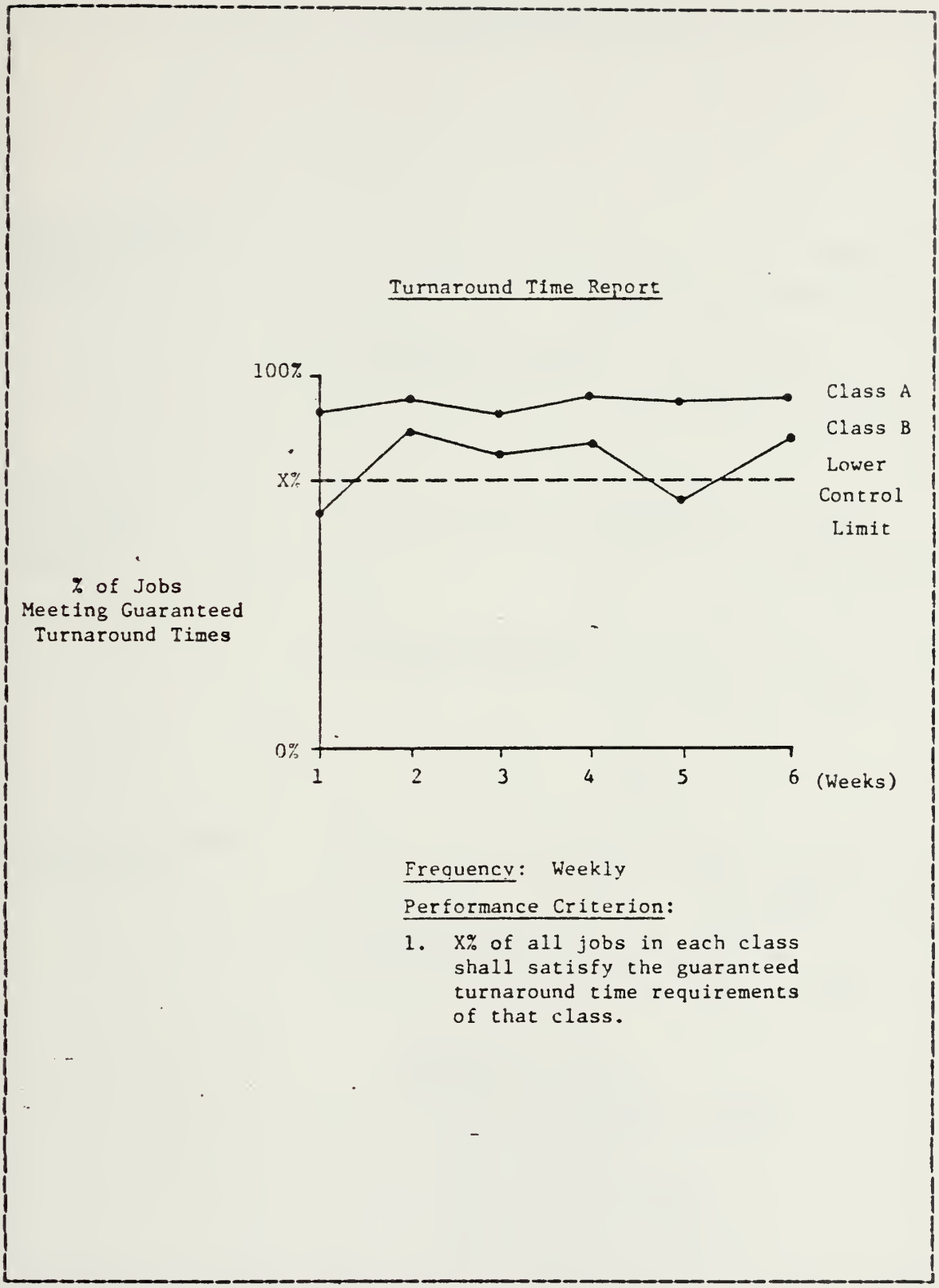
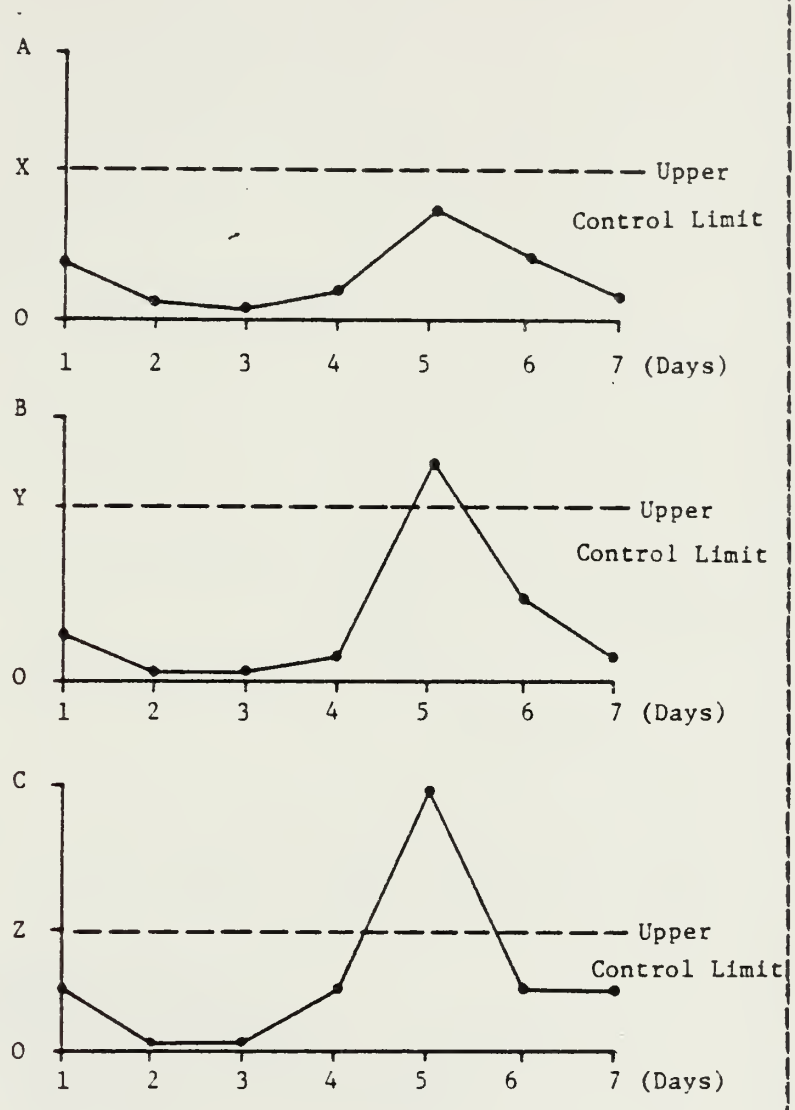


Figure 3.2 Turnaround Time Report.

System Reliability Report



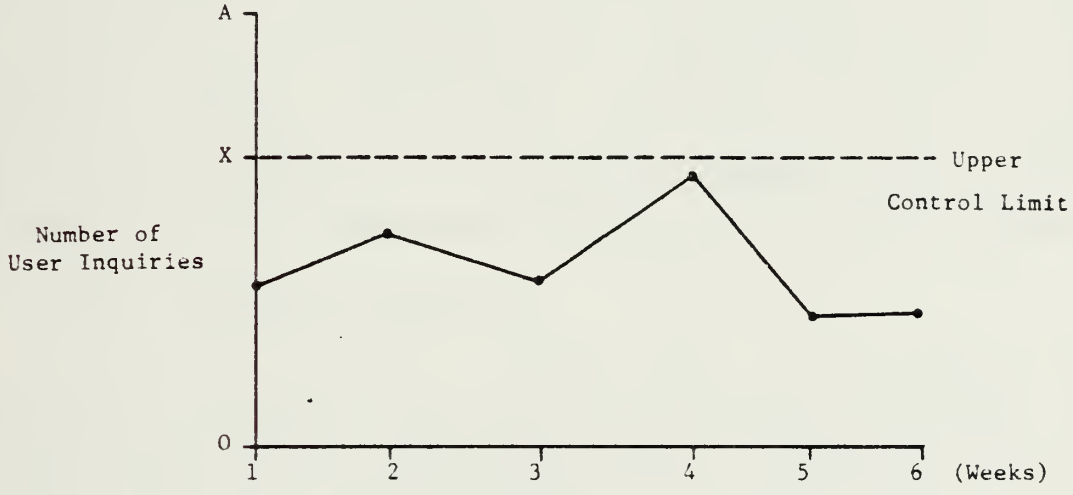
Frequency: Daily

Performance Criteria:

1. The average length of system interruptions shall be less than X minutes.
2. No interruption shall last longer than Y minutes.
3. There shall be no more than Z interruptions per day.

Figure 3.3 System Reliability Report.

User Support Report



Frequency: Weekly

Performance Criterion:

1. The number of inquiries to the user support group shall not exceed X per week.

Figure 3.4 User Support Report.

management of a CPM program. Periodic review of the CPM effort by CPM management should be performed to insure that the basic goals of the CPM effort are still in focus.

a. Define Scope and Objectives

As mentioned earlier, the CPM management should develop and have a clear understanding and definition of the scope of its responsibilities. Next, the proposed CPM program objectives should be established and the information requirements should be defined in order to insure that only pertinent data will be collected during the later phases of the CPM program. Finally the nature of the actual reports to be produced, the frequency with which they should be put together, and the performance criteria related to each should be determined. It is important to mention again, that the reports should be clearly presented and contain only meaningful and pertinent information for the readers.

b. Determine Approach

The critical resource for a successful CPM program is not modern, up-to-date measurement equipment, but skilled, knowledgable personnel who are intimately familiar with the computer resources being measured and the tools being used, and who have the analytical ability to appraise and interpret the measurement data. As mentioned earlier, without such a resource and a well conceived CPM program, the return on the investment in time will most likely be disappointing if not disastrous. Two possible sources exist for these key people: current (or future) employees, or consultants from outside firms.

The former of the two sources is preferred since it allows for better project control, and has the additional advantage that the in-house personnel are more aware of the objectives and internal workings of the organization.

Some advantages of the outside consultant approach are its cost-effectiveness (for organizations with no available performance measurement resources--i.e., personnel, monitors, etc.), its objectivity, and the opportunity for knowledge transfer from the consultants to in-house contacts. Possible disadvantages of contracting out include the consultants' lack of familiarity with the internal operations of the organization, and the possible friction that may result between the consultants and in-house personnel.

c. Control and Review

"The failure of many performance improvement efforts has been traced to the lack of genuine management interest and support" [Ref. 4]. A probable cause of this is having a CPM group that is not totally dedicated to the CPM effort, in that CPM is not their prime responsibility. They become spread "too thin" and if CPM has a low priority, it suffers the consequences. Again, this problem can be solved by making sure that at least one member in the CPM mamagement group has CPM as a top priority and this member can dedicate full time to the CPM program. Another possible cause in the decline in interest in the CPM program can result from false hopes and unrealistic expectations for cost savings, especially if major expensive measurement resources like hardware monitors must be purchased. The initial costs of a major CPM program can be well over $100,000 in hardware resources alone.

The total CPM program should be reviewed periodically to insure changes in informational need are reflected in new CPM reports. Existing reports should be examined to determine their current relevancy; irrelevant reports are often generated when the need for them has long since passed.

3. Resource Requirements

Many of the resources required for CPM have been mentioned along with their cost. Perhaps the most significant costs of any CPM program are those experienced during the start-up phase of the effort. Such costs usually include: the program's initial planning, system modifications to acquire the necessary data, acquisition of commercial products and packages, and the development of report generation mechanisms. Once underway, continuous reporting demands other costs and resources: system overhead to collect performance data, processing time to analyze the data, and manpower to interpret the data and maintain the reporting system. In addition, in-depth studies prompted by exception reports and an intolerable decrease in performance require the use of rather sophisticated tools--notably, hardware and software monitors and modeling programs. The cost of these tools varies with the type and accuracy of data to be obtained. The additional costs of personnel and training may, once again, make it more cost-effective to hire outside consultants.

IV. CPE TOOLS AND THE SYSTEM LIFE CYCLE

As mentioned earlier, the two most important and effective tools for evaluation are visual inspection and common sense. The use of other tools without these will most likely be ineffectual. The additional CPE tools avaliable include:

- Accounting Data Programs
- Program Optimizers
- Software Monitors
- Hardware Monitors
- Benchmarks
- Modeling

CPE tools generally fall into one of two broad categories: measurement or predictive. Accounting packages, software and hardware monitors and source program optimizers fall into the measurement group. Modeling is a prediction tool and benchmarks are a point of reference for measurements and don't really fit totally in one or the other of these categories. These tools and their use will be described as related to the life cycle of a computer system.

A. CPE TOOLS

1. Accounting Data Reduction Programs

The need for an accounting program grew out of the requirement to fairly distribute the cost of the services provided by a computer system amongst it's users. Today, most systems come with accounting programs having varying

degrees of capability to gather information about the system and individual programs. In early multiprogramming environments, it was noticed that essentially identical runs could generate a wide range of charges. This caused managers and users to search the details of the accounting data collected and to develop ways to collect more detailed information in order to uncover the reasons for the charge variations [Ref. 1].

System accounting packages generally collect three types of information: identity, quanity, and time. Identity data includes such things as program name, charge code, device number and type, and other information that is useful for categorizing accounting data like programmer name or user name, type of run (test or production), priority of run, etc. Quantity data involves the amount of something used by a resource such as the amount of prime memory requested or used, total secondary storage space used, count of I/O requests to a disk or tape, number of pages printed, etc. This data can be at a very detailed level, i.e. the actual number of logical records and both the block count and the number of records in a block. Time data can show the amount of time used by a job or user for the different resources of the system.

Accounting packages produce reports which show processor time use, total memory requirements of the workload, and give all the statistics collected in a summary format for review. Modifications to the accounting package programs and the development of new programs can provide user/job utilization reports.

Accounting data can point to those programs that use the most system time and resources. These programs are prime candidates for increased system performance through improved program code and resource utilization. Morris [Ref. 1], gave examples of this in the chapter on accounting

data. In one example, it was discovered that a disk file, used by a program requiring a great deal of I/O time, used unblocked card image records. When the records were blocked to track buffer size, the program executed twice as fast and used considerably less secondary storage. Another example involves a program using multiple files that were stored on the same disk. By placing certain files on different disks, device and channel contention was reduced which resulted in increased effective utilization of the system resources. This is a case where the hardware may have been performing efficiently but the system resources were not being used effectively.

Accounting data can also be used for effective job scheduling. Through detailed studies of resource usage of major jobs a more optimum job schedule 'mix' can be achieved that can increase system performance, especially in an active multiprogramming enviornment.

Accounting packages are the most widely used CPE tool throughout the industry. Development of more comprehensive accounting programs has lead to more sophisticated and diversified applications of the accounting data in terms of system performance evaluation. Morris [Ref. 1], says that "One of the most promising developments of the CPE field that has largely been made practical by the availability of acconting data and suitable reduction packages is Kolences' Theory of Software Physics" [Ref. 10]. According to Morris, this work provides a completely rigorous definition of software "work" and "power" in a modern computing equipment context and applies the physics-like terms and definitions to practical applications in billing and charging for system usage as well as in the more encompassing field of computer system capacity management. The text by Kolence is suited to individuals with an interest in the full potential of accounting data and is one possible

theoretical basis for an all-encompassing description of computer system performance.

2. Software Monitors

Software monitors are simular to accounting programs but collect much more detailed information. They can make measurements at the step-by-step instruction execution level. Software monitors are programs that are attached to the operating system and capture event and timing data directly from internal tables setup and maintained by the operating system. Data sample rate can be varied by the user and captured data is stored on tape or in disk files.

Software monitors are operating system/hardware dependent. Even a modest operating system change can impact a software monitor program. Software monitors usually fall into one of three groups: sampling, time-driven or event-driven.

Sampling monitors extract a "sample" or subset of the entire population of interest and through the use of statistical methods inferences are made about the entire population. Variation of the sample size provides a range of confidence intervals to suit the accuracy requirement for a given measurement. Event-driven software monitors rely upon "hooks" which are inserted into the code to be monitored or by a detection of a change of state. A hook is simply an embeded instruction that triggers the collection of predetermined data by the software monitor. Data collection can also be triggered upon the completion of "n" occurences of a hook if needed in order to prevent excessive monitor overhead or "artifact" from dominating the program being measured. A change of state is defined as the transition from one activity to another activity. An activity can be a change from privledged to unprivledged use of the CPU, the change from CPU wait to CPU busy, etc. Changes of

states generally occur less frequently then do hooks and therefore performance data is usually collected at each occurence. Time-driven data collection uses a clock or timer and at fixed intervals of time collects measurement data by interrupting the execution of the program under measurement. The most capable software monitors use a combination of these three types of data collection and selection methods. Software monitors use system resources (i.e. CPU cycles, memory, etc.) that must be shared with the program or programs that are the subject of the evaluation effort and can therefore "contaminate" the analysis by its very existence.

3. Program Optimizers

This tool is a subset of both accounting packages and software monitors. Actually, the program optimizer does not directly optimize code, but rather it points out areas in the program that would produce the most benefit from programmer provided optimization. Since program optimizers are compiled together with the program of interest, they are computer dependent. Program optimizers or "analyzers" can determine those parts of a program with the most activity. These are called the "critical part" of the program and typically "comprise from 1% to 10% of all program statements in a job or program". [Ref. 11] Unfortunately, program optimizers are commercially available for only two major high-level languages, FORTRAN and COBOL [Ref. 1].

Application software is not the only software of interest when considering factors that influence system performance. System software (operating system, utilities, etc.) also use and consume valuable resources and should not be excluded from analysis consideration. Control programs analyzers (CPA) help to analyze this class of software. In some operating system development, the only software

guidelines are that it works. How efficiently and effectively it works are of secondary importance. A later section will discuss in detail the improvement of system performance through improved software practices in the critical section of a program.

4. Hardware Monitors

Hardware monitors are external electronic devices that are attached though the use of probes, to the various internal connections of a computer system to sense and record changes of state in the components. An intelligent hardware monitors not only have the capability to probe the hardware but can also be connected by a line of communication to the measured computer system. The intelligent monitor incorporates its own computer, complete with monitoring software and uses this line of communiction to actually control the monitoring process via coordinated communication between the monitor and the measured computer system.

Although hardware monitors are becoming "easier" to use, they are still the most demanding in the area of technical knowledge required, especially in the area of system architecture and implementation down to the circuit level and require the largest financial investment. In addition, hardware monitors can be easily misused or incorrectly used and can generate data at such a detailed level and in such large amounts that inexperienced users often find themselves overwhelmed with data after only a short period of data collection. However, their low level of detailed information gathering capability, miniminal operating overhead and the fact that they are generally system independent, makes them an attractive and desirable monitoring tool. The choice of selecting between a hardware or software monitor will be discussed in greater detail in a later chapter.

5. Benchmarks

Benchmarks are programs that are designed to specifically represent the workload or the set of workloads that exist or are anticipated to exist on current or proposed future computer systems. Benchmarks act as a reference unit of measure to be compared with the results produced by other CPE tools.

A small set of benchmarks that have been validated as accurately representing the total workload at a given period in the history of the system can be extremely valuable to a CPM program. Performance data produced by periodic executions of benchmarks allow the CPM group to evaluate changes made to the computer system. This periodic comparision process provides data to easily chart the positive (improved performance) and negative (degraded performance) impact made on the system.

6. Modeling

Modeling is a discipline in its own right and has been widely used thoughout many fields. Modeling makes use of mathematical descriptions to provide an analytic or simulated model of the system to be studied. Modeling in the computer enviornment allows the detailed examination of a computer system or set of programs before they actually exist. In this sense, models allow the "nature" of something to exist without the acutal existence of that thing.

The most important aspect of modeling is that it allows the question "what if . . .?" to be asked in a variety of different ways through the use of parameters, with the model producing results for each of these different requests. Modeling can also provide the results of applying a proposed solution as suggested through the use of other CPE tools, to a problem without actually implementing the

solution suggested. For example, a model could try different configurations of memory, channels and disk units without ever installing the actual devices. This feature of modeling is extremely powerful, expecially in long range CPM/CPE efforts. However, a model must be rigorously validated and if possible verified with existing sytems. Developing a model is very time intensive and therefore very expensive. For a more detailed discussion of these tools see Morris [Ref. 1], in which a chapter (3-9) is devoted to discussing each tool.

B. CPM AND COMPUTER SYSTEM LIFE CYCLE

It is highly desirable to develop a CPM program that goes beyond the operation phase of the computer life cycle and encompasses the complete life cycle from procurement phase to transition phase. Although not suitable for all computer facilities due to costs or other reasons, the following section lists some ideas to consider during the life cycle of a computer system.

1. Procurement Phase

This phase of the life cycle can be the most important, in that the framework for the potential expansion of the computer system is decided. This decision obviously has the greatest impact on all the following phases. It is at this phase that the modeling tool is most powerful and most enlightening. After arriving at a workload specification, modeling can be used to allow alternative parameters to be evaluated for such resources and components as the type of peripheral device best suited to a particular file structure or configuration of memory that will be needed. Ranges of relative speeds and capacities can be computed from the model that will allow selection to begin on major equipment

such as CPU, memory and supporting equipment such as tape drives, disks and printers. Thus, the understanding of the proposed initial and future anticipated workload and equipment will aid in the selection and development of benchmarks of the system. Benchmarks will play a major role in preparing the requests for proposals (RFP), also known as procurement documentation, for the pending equipment selection.

Vendors, once aware that a perspective customer has developed a model capability and has selected benchmarks for a system, are more apt to become competitive by making modifications and custom tailoring their proposal to meet the customers needs more closely and at the best price. "The proposal review is one of the most important steps in the system's life cycle and in the CPE effort because it is the one time when all potential suppliers are most intensely interested in the buyer's needs" [Ref. 1].

At a recent Computer Performance Users Group meeting an example was given by a speaker, in which inadequate attention was given to matching the new system with the anticipated workload involved a government agency that purchased a computer costing less than $300,000. Having procured the new system, the agency proceeded to develop several million dollars of software to be run on it. The one system soon proved to be inadequate to accomodate the workload, and several more were eventually purchased. Difficulties arose trying to interconnect these computers, and after the purchase of five identical systems and connecting them together the system still was unable to provide sufficient service to handle the workload at a satisfactory performance level. This situation would be much less likely to have occured if a CPM program had been in effect during the procurement phase. However, it is thought that many government organizations are forced to buy

inadequate computer power as a result of governmentation computer procurement procedures. Larger system procurement require more paperwork and have longer procurement times. This may force some organizations to buy a smaller, less adequate system based upon the amount of procurement paperwork required instead of the ability of the system to handle the workload.

The selection or final step in the procurement phase relies less on performance factors since by this time only proposals that should be considered have provided positive benchmark results, and the remaining selection criteria is price, delivery schedule, vendor support, conversion costs, etc.

2. Installation Phase

The two most important concerns in this phase are: establish the system configuration that is best tailored to the workload, and verify that the delivered and installed equipment does indeed perform as per written specification. The former concern means going over any and all workload characteristics again and verifying that the equipment and configuration is correct. This is the last chance to make sure everything is "right". At this phase the buyer has the upperhand in that the system is not "signed off" yet and the vendor will be motivated to be most accomodating at. Modifications that can be simply made at this phase become increasing more difficult and expensive at later phases. The latter concern, that actual performance equals or exceeds advertised values, should be demonstrated by the vendor.

It is at this phase that CPM/CPE personnel can learn a great deal about the new system directly from the vendor installation team. The buyer should get involved, tactfully "buy lunch", learn how to use the system diagnostics, and

generally learn everything that they can from the people installing the system. If the relationship between the buyer and vendor is positive and accepted by the vendor, it is more likely that the vendor technicians will point out both strong and weak points in the system. The CPE personnel can gain extremely valuable information as a consequence of this relationship in that detailed technical information relating to performance evaluation such as system table addresses, pin location, probe points, etc. can be obtained from current as well as potential future contacts.

3. Operations Phase

It is during the operations phase that most people consider some sort of CPM or CPE effort. As has been shown, many potential problems reguarding initial and long term performance can be avcided or solved prior to this phase through the use of a CPM program and CPE tools. Even with all of the prior CPM efforts, a dynamic and expanding workload will require continuous attention from the CPM group. Each new program should be reviewed as to it's impact on the workload and performance of the system. Accounting data along with supporting performance data gathering programs provide sufficient information to assess the impact. If problems are uncovered, more sophicated CPE tools may be required to pin point the appropriate solutions.

If a model does exist that closely resembles the system, it may be able to determine the benefits and advantages of new products and options that become avaliable throughout the life of the system. Through the use of a responsive and cost effective CPM program and CPE efforts, the system's point of saturation can be substantially delayed. Thus putting off a considerable expenditure for a major upgrade or new system.

Once again, a system model is very useful and valuable in determining the capability of the existing system to handle an expanding workload and to reveal what or where system changes and modifications will be required in order to satisfactory maintain this workload. The limit of this process is system saturation, where additional enhancements to the system are not possible or not practical or not cost effective. It is at this time that the computer system will move into the transition phase of it's life cycle.

4. Transition Phase

At this point, the CPM group should have gained a great deal of experience, become expert with the dynamics of the organization's workload growth characteristics and expert on a system that has managed to adequately process the workload until this time. Along with an indepth personal knowledge of the installation's performance history, the CPM group should be well prepared to aid in system transition once it has been predicted.

An organizatien that has effected and benefited by a CPM philsophy and program should be in a very good postion to make an orderly and intelligent transition in which major, unanticipated performance problems will be minimized, if they occur at all.

The above discussion of CPE tools provided some very general guidelines in the use of these tools during the four phases of the life cycle of a computer system. However, two CPE tools that seem to be used most often throughout the life cycle CPM effort are accounting information and system and workload modeling. Due to the low cost of accounting packages it may be desirable to implement the most extensive and detailed one avaliable. It should be an ongoing effort to become increasingly knowledgable about the operating system software so that accounting programs can be

supplemented with additional capability in order to supply all the information that is desired in the CPM process. The source code of the accounting system is an invaluable document in this regard, especially in answering the "where" and "how" questions in obtaining the data.

As useful as models seem to be, they are extremely hard to develop and validate to any degree of accuracy in imitating the actual situation. However, it is suggested that, as a minimum, a resource or component diagram be constructed of the system showing capacity, speed, buffer size, transfer rate, etc. of each component, as well as lines of communications and/or channels and their associated rates and capacities. Some simple calculation using these values can be very informative when trying to match devices and resources. The same idea could be followed for the workload where all the programs would be listed along with the resources and amount required as well as critical time values, such as response time. Even the simplest of models can be quite helpful for example, when components (and associated speed, capacity, etc.) of a system are listed in a group as well as a description of will be running on the system. This "paper" model can provide a feel for potential problem areas such as mismatch of the number of channels for a certain number of devices, or that the amount of memory presently configuared will only allow a very limited multiprogramming enviornment to exist. Table I shows the primary and secondary tools required throughout the life cycle process as proposed by Morris [Ref. 1].

C. PRACTICAL ARCHITECTUAL ENHANCEMENTS

Svobodova [Ref. 12], suggests that the utility and simplicity of the following architectual enhancements would make them practical to implement.

TABLE I

CPE Tools and System Life Cycle

PHASE	ACTIVITY	MODELING	BENCHMARKS	ACCOUNTING DATA	SOFTWARE MONITORS	HARDWARE MONITORS	PROGRAM OPTIMIZERS
	CPE TOOLS						
Procurement	Conceptual Design of Workload	P					
	Detailed Workload Specification	P	S				
	Equipment Specification	P	S				
	Request for Proposals (RFP)	P	S				
	Review of RFP Responses	S	P				
	Select Equipment Supplier		P				
Installation	Tailor Workload to Equipment	S	P	S	S	S	S
	Configuring Equipment	S	P	S	S	S	
	Installation and Checkout		P			S	
	Acceptance Testing		P	S	S	S	
Operation	Workload Implementation		S	P	S	S	S
	Program Reviews	S		S	P	S	S
	Configuration Enhancement	S	S	P	S	S	
	Adding New Workload	S	S	P	S	S	S
	Projecting System Growth	P	S	S			
	Predicting System Saturation	P	S	S			
Transition	Review of New Systems	P	S				
	Review of Potential Data Processing Needs	P					
	Reuse Analysis of Owned Equipment	P	S				
	Conceptual Design of Foreseeable Workload	P					

P = Primary Tool
S = Supporting Tool

1. System Timer. The system timer is a standard feature of many prsent computers. Accuracy of timeable measures monitored by a software monitor is limited by the resolution of the system timer. The system timer can be used in one or both of its modes:
   a) Data mode: The system timer provides time stamps for monitored events (time of day clock).
   b) Control mode: Interrupts generated when the timer expires drive the sampling routine of a software monitor, or synchronize a hardware monitor with software activities of the measured system (interval timer).
2. Instruction counter. The instruction counter[1] counts executed instructions, either totally or conditionally (instructions executed in problem state, instructions executed in the partition N, etc.).
   a) Data mode: The instruction counter facilitates measurement of parameters such as instruction execution rate, instructions executed per I/O, CPU utilization.
   b) Control mode: Similar to timer interrupts, the instruction counter can generate interrupts that are used for the purpose of program sampling.
3. Memory cycle counter. The memory cycle counter counts memory references rather than executed instructions.
   a) Data mode: The memory cycle counter facilitates measurement of parameters such as memory reference rate, memory references per instruction, memory utilization, memory references between page faults.

---

[1]The term instruction is sometimes used for a register that holds the address of the next instruction to be executed (program counter, instruction address register). In this context, we are using the term instruction address register.

b) **Control mode**: The memory cycle counter can generate interrupts that drive a monitor routine sampling program for the purpose of examining the locality of reference.

4. **Address match circuit**. Many computers have a hardware debugging aid, the address match circuit. The address to be matched is loaded manually thourgh toggle switches. A match generates a pulse in the CPU circuitry, but this pulse is not recognized by software, which puts a constraint on the utility of such a feature as a monitoring aid. To become a monitoring aid, the address match circuit has to be given interrupt capabilities. A special instruction can be provided to load the address match register under program control; the absolute address that is to be matched then does not have to be known explicitly. A match pulse causes an interrupt that transfers control to a monitoring routine. The address match circuit thus provides a remote hook. Together with an associative memory of sufficient size, the address match circuit enables system software level instrumentation without any physical changes to the monitored software. Such instrumentation could be temporary (for testing software monitor routines), or permanent (if for some reason the monitored software must not be modified). In addition, match pulses are potential synchronizing signals for a hardware monitor.

5. **Monitor call instruction**. The monitor call instruction provides access to system monitor services. Such a feature has been implemented on the IBM S/370. The 'Monitor Call' instruction (MC) operates similarly to the IBM Supervisor call (SVC). A special control register, called the monitor register,

functions as a mask for different monitoring classes in the same way a mask is used with system interrupts. A special privileged instruction, 'Set Control Register' can change the mask. A 'Monitor Call' instruction then designates one of the sixteen monitoring classes. If the designated class is unmasked, then execution of this instruction causes an interrupt that is processed by the operating system software.

6. Hardware monitor plug interface. The function of this interface is to concentrate signals representing most frequently measured hardware events, and stabilize these signals for the period of event duration. The interface is placed in a location that is easy to reach, with terminators that provide an easy connection for a hardware monitor.

V. CHOOSING A MONITOR-HARDWARE OR SOFTWARE?

If the decision has been made that the accounting data no longer provides the level of performance information required by the CPM group, then the next logical decision would be to choose between a hardware monitor or a software monitor. The selection process reduces itself to picking the monitor whose characteristics most closely satisify or meet the user's requirements. A very informative article which addresses the monitor selection issue is "Performance Monitors-Hardware or Software?" [Ref. 13]. The following provides some of the main ideas regarding the question of selecting a monitor.

A. SOME GENERAL CONSIDERATIONS

There exist at least three considerations common to both types of monitors:

- **Cost effectiveness** -As a CPM program grows, the issue of how cost effective the next step is, should always be considered. Purchasing either monitor type will demand the allocation of more resources, both financial and personnel. A monitor will incure the following costs: the monitor itself, personnel (both time and training) and most likely the use of the existing computer resources for performance data reduction and report generation and special experiments. The initial cost of a monitor can range from several thousand dollars to well over one hundred thousand dollars. All of these costs must be totaled and compared to the anticipated benefits from taking this step. The net result will determine if it is cost effective.

• **Definition of objectives** -As with the CPM program, a plan is needed for every important CPE endeavor, especially the purchase of a new product or resource. Issues involve: how the monitor will be used, what impact it will have on the system, what will be measured, etc.

• **Personnel capabilities**--This issue has been discussed earlier in this paper. The important consideration here is that the monitor will require properly trained and qualified personnel to use it and these personnel must be able to communicate with vendor support personnel.

1. Characteristics of Hardware and Software Monitors

Table II presents a quick reference for comparing and evaluating twelve characteristics of the two types of monitors [Ref. 13]. An elaboration of each of these monitor characteristics follows:

1. **Cost**--Since a hardware monitor is a combination of hardware (some have full capability microcomputers) and a software package, it's initial cost can be moderately, to considerably more than that of a software monitor. Operating costs of a software monitor are considerably lower than those of a hardware monitor, since its operation and configuration is relatively inflexible, and while set-up time for a hardware monitor can be quite extensive (finding and attaching probes, configuring logic boards or programs, etc.) it is minimal or not required (y the user) for a software monitor. Formal training in the use of a hardware monitor as well as the design of experiments also adds to the cost of using a hardware monitor more sc for using a software monitor.
2. **System overhead**--Hardware monitors which only use probes add no additional overhead to the system being

TABLE II

Characteristics of Hardware and Software Monitors

Characteristic	Hardware	Software
1. Cost	High	Low to medium
2. System overhead	None	Variable
3. HW/OS dependent	No	Yes
4. Precision	Monitor depend.	Host system depend.
5. Capture of qualitative info.	Limited	Yes
6. Training reqd.	Extensive	Limited
7. Flexibility	Good	Poor
8. Ease of use	Poor	Good
9. Portability	Good	can't (unless ident. system-then good)
10. Device monitoring	Yes	No
11. Interrupt	None	Yes
12. Expected life	Good	Fair

monitored. Hardware monitors that have communication links with the monitored system can impose some overhead. Software monitors can inflict substantial overhead on the system since they are programs which are run on the monitored system and require the use of the system's resources such as CPU, primary and secondary storage and I/O services. The severity of the overhead, also referred to as "artifact", is dependent upon the rate at which sampling or measuring is being performed. With the exception of some intelligent hardware monitors, both monitors rely upon the measured system's resources for data reduction and report generation. Some real-time systems may not be able to tolerate even the slightest amount of additional overhead imposed by a monitor, and would require a probe attached hardware monitor.

3. **Hardware/operating system dependent**--With the exception of some intelligent hardware monitors, hardware monitors are essentially independent of both the hardware and software of the measured system. In some cases inadequate or inaccessable pin locations can be a very serious problem, preventing full use of a hardware monitor. Software monitors use system specific information and are therefore custom designed for a particular hardware/software configuration for a particular brand of computer. Software monitors don't exist for all computers whereas, if performance related probe points exist on a system, a hardware monitor can be hooked up to it.
4. **Precision**--Precision in a hardware monitor is dependent upon the following monitor internal components: the resolution of the clock, speed of it's counters, accumulators and other internal circuitry. Therefore, it is desirable to use a hardware monitor

with a precision greater then that of the system being monitored. Since software monitors utilize the monitored system's internal clock and the ability of the operating system maintained measurements, it's percision is usually lower then that of a hardware monitors potentical capability. Due to variation caused by the sampling technique typically used by software monitors, precison can drift. For the majority of measurements, either monitor will prove to be satisfactory. In those few cases in which extreme precision must be used, hardware monitors are necessary.

5. **Capture of qualitative information**--Software monitors can more easily explore qualitative information such as the contents of main memory, queue contents, program ID's, and other descriptive variables stored in tables. The ability of most hardware monitors to explore qualitative values is restricted by the hardware connections. Some intelligent hardware monitors do permit some qualitative information to be transferred to the hardware monitor via the use of it's communication link. However, the ability of hardware mcnitors to collect qualitative information is limited relative to that of software monitors.
6. **Training**--Hardware monitors, due to their very technical nature, require more training which is critical to the degree of success attainable. Hardware monitors must be configured by the users which requires selection, location and varification of probe points, wiring of logic plugboards, etc. In as much as the hardware monitor is system independent, it is extremely user/analyst dependent, and the better trained the user the more benefit can be gained from the monitor's capabilities. Software monitors

require relatively little training for implementation and operation. Software monitors have a "fixed" set of reports and are more easily analyzed than are reports from a hardware monitor.

7. **Flexibility**--Measurement flexibility potential exists in both monitors. However, to achieve any degree of measurement flexibility in a software monitor, modifications must be made to the code in the data collection program, the data reduction program and the analysis and report generating programs. Short of performing these possibly extensible modifications, the software monitor remains relatively inflexible to change of the types of measurements. On the other hand, the hardware monitor is designed to be very flexible. In general, anything can be measured for which a probe point exists, although data reduction programs may be limited to only fairly standard measurements and require modification.
8. **Ease of use**--Compared to hardware monitors, the software monitor is very easy to use. Activation and selection of desired reports is usually requested though the input of simple console commands. Hardware monitors can take several days of set-up planning, coordination with the system users in the area of scheduling, and can lead to disruption, confusion and interference in the computer facility. Each new measurement which uses different probe points requires reconfiguration and planning.
9. **Portability**--There are two major divisions relating to portability: amongst different brand name systems and amongst same brand name systems. The hardware monitor is extremely portable in either division. All the hardware monitor depends upon is the availability of attachable probe points. Software

monitors are only portable amongst systems that have the same hardware/software configuration. Since software monitors are relatively inexpensive compared to hardware monitors, it may be as cost effective to buy a few software monitors for dissimilar systems as it would be to buy one expensive hardware monitor. One advantage of multiple software monitors over a single hardware monitor is that the software monitors can run simultaneously and uninterrupted. A hardware monitor can only be active on one system and must be totally reconfigured when moved to another system. One other consideration is that hardware monitors are much more susceptable to "handling" problems in that they are delicate electron devices, and could be easily broken.

10. **Device monitoring**--Hardware monitors may be attached to any device having a connector. This type of attachment provides a different "view" of the device then that possible through the use of a software monitor in that hardware monitors "stand at the door" to review data entering and leaving whereas a software monitor can walk in and freely explore the contents of the room. This is true in the case of a device such as a disk.
11. **Interrupt considerations**--Software monitors can be adversely affected by interrupt conditions in the CPU in that they can be locked out, or they can take advantage of the interrupt capability to stop processing and allow the software monitor to explore any part of the system. Since most hardware monitors are external, they are not affected by interrupts.
12. **Expected life**--Hardware monitors can be exected to remain relatively useful for a long period of time if connection pins remain avaliable. Software monitors

are dependent upon a specific configuation and minor changes to this configuration can impact the useful life expectency of a software monitor.

B. MAKING THE DECISION

As mentioned in the beginning of this section, the selection of a monitor is primarily a question of matching the specific needs of the CPM group with the capabilities of each of the products avaliable. The preceeding twelve points should form the basis for making the comparison of what is avaliable. The importance of initially defining objectives can not be over stressed. The selection process is more apt to be successful if it is known beforehand exactly what is to be measured, what questions are to be asked, and what desired end result is expected. In some instances, circumstances may exist that leave no choice but selection by default. A software monitor may not exist for a particular computer system, or a real-time or highly active multiprogramming system may not be able to tolerate additional overhead created by a software monitor, so a hardware monitor is the only choice by default. In unique applications, such as the space shuttle program, special measurements may have to be made requiring the minimum overhead of a hardware monitor. Since software monitors are by far the less expensive of the two, organizations that have limited budgets may have no choice other than to buy a software monitor. Shortage, or nonexistence, of personnel possessing the required backround or general experience necessary to be trained in the use of hardware monitors may also lead to the same choice. The case studies found in Bookman [Ref. 14], "Saving from Performance Monitoring" [Ref. 15] and Sewald [Ref. 16] provide ways which other organizations have elected to choose a monitor and conduct measurement programs.

VI. SOME COMMON SENSE PERFORMANCE FALLACIES

When faced with a problem, most people will either have or will soon have a "hunch" or an intuitive explanination or hypothesis as to what is causing the problem, or what it may take to make the problem go away. In almost every case, extensive remedies should not be implemented until "proven" to be the correct or at least the best possible one. In the context of computer performance evaluation, the best method for proving a hypothesis correct is through careful measurement. If an intuition leads to the implementation of a remedy without adequate verification and proof, considerable time and money can be wasted. Therefore, one must be careful about the use of common sense guidance about where bottlenecks might be in the system and what causes them, since it is very easy to be mislead in such a complex system.

This chapter will discuss some specific, possible, and general fallacies of intuition as noted by Carlson in his article "Fallacies of Intuition in Performance Tuning" [Ref. 17].

A. SOME SPECIFIC FALLACIES

1. **Fallacy: During peak load processing the CPU busy increases significantly.**

This is a carry over from the old days when operations people would judge the behavior of the CPU by the activity of the noisy peripherals. Actually, the peak load processing can have very little impact on actual CPU cycles used, but can have significant impact on noisy peripherals. Additionally, in many cases it was found that the "CPU usage

would increase only 2% to 4% in CPU busy" [Ref. 17], however, a peripheral like the printer would become twice as busy. An increase then in peripheral activity would give the impression that the whole system was much more active.

Not surprisingly, and without verification, many computer facilities would seek to increase the capacity of their system. This normally resulted in a desire for more CPU power or additional memory. Obviously in cases where the actual increase in CPU activity during peak workloads is very small, modifications to the CPU would have little or no effect on the overall performance of the system. What is needed in this situation is a careful balancing of the peripheral equipment based upon the results of resource utilization measurements, and a possible review of accounting data to come up with a better job scheduling algorithm.

2. **Fallacy: The reason for slow terminal response times is slow disk access times.**

Some simple comparisions of only time and rate values for cooperating devices and communications links could lead to the above conclusion. Terminals are attached to controllers (which have fast internal speeds), that coordinate CPU's that operate in microseconds or nanoseconds, and disk units which operate in milliseconds. Therefore, it would seem intuitively sensible that the disks, which operate 1000 times slower than CPU, are the most probable cause for delay. Without going any further than the comparisions of the above information, it seems that obtaining faster access secondary storage devices would improve response time.

Carlson devised plan to measure those activities that influence response time. A hardware monitor was selected to record all major events occurring during response time delay. Factor analysis, a statistical technique, was used to indicate which variables might be influencing response

time the most. The results appear in table III and show that disk activity accounted for only 14% of the response

TABLE III

Response Time Components

Activity During Response Time	Percent of Response Time
Terminal Controller	21%
Disks Only Busy	14%
Channel busy	5%
TP Supervisor Program	15%
LCS Inhibit	18%
Non-interruptable	46%
Logical OR of all Items	94%
Unaccounted for	6%

time delay. Replacing the old disk with one having one-half the access time (twice as fast) would only result in a 7% reduction in repsonse time. If the access time were to go to zero, the net result would be only 14% reduction in access time. In this operating system (IBM OS) overhead CPU functions (i.e. non-interruptable activity) accounted for almost half, or 46% of the response time delay.

This example from Carlson [Ref. 17], points out that there are rarely one-to-one relationships between different devices in today's complex computer systems. The fact that a disk drive operates one thousand times slower than the CPU has only a small impact or minor relationship to overall system performance. The computer system is much more complicated than a one-to-one additive relationship. Therefore, in this case it was not so much the performance

of the equipment or what equipment was used, but rather how things were done-i.e., the system software. A close look at tuning of the interrupt and I/O handlers could bring about much greater reductions in response time.

3. **Fallacy: Switching from single density to double density disks will cause system delay due to arm contention.**

Doubling the density while keeping the arm mechanism the same can be viewed as potentially doubling the demands on the access arm. In view of this, it is easy to see how the above intuitive belief could be supported. Further support for this fallacy stems from the fact that one would expect a fairly high number of concurrent seeks to occur due to the highly active operating system that must support a multiprogramming environment and disk controllers that support multiple disk units. The fact is, that going from single to double density has a negligible impact on overall performance. Careful analysis of many operating systems indicate, however, that only rarely are concurrent seeks attempted on any of the packs on a given controller. The following example was found in Carlson [Ref. 17]. A hardware monitor was used on disk drives, clustered eight to a controller. Pins were located that were thought to provide evidence as to whether or not a seek was in progress on any of the packs. The initial measurements seemed to reveal that seeks were in progress 100% of the time, indicating that the disk controller was extremely busy. However, it was soon found out that the pins being probed were only indicating if power was on in the disk controller. This points out the very important need to verify that the pin does indeed provide the desired measurement information. One very simple, although not conclusive method to validate the pin, would have been to push the halt button on the system and notice a corresponding halting of disk activity indicated in the collected data.

The correct pins were located and verified and it was found that on a typical 8-pack disk subsystem, overlapped seeks account for less than 1% of the time. The maximum seek overlap ever measured was only 3%. The experiment was conducted prior to and after the change from single to dual density packs. The system was stabilized with single density drives and measurements were made of CPU activity, channel activity, response time, etc. Then the most active disk pack was combined with the least active disk pack on one dual density drive. The same measurements were made again with no measurable difference in behavior. Even combining the two most active packs on one double density pack revealed no perceptable degradation or change in system activity as indicated by conducting the same test. Thus, careful measurements seem to dispute the intuitive idea of increased access arm contention as a result of switching from single density to double density disk packs. Therefore, the decision could be made to go to double density packs based upon economic issues, in that dual density disks may provide lower cost-per-bit secondary storage.

4. **Fallacy: Large core storage (LCS) or asynchronous memeory is always a good thing to put on a system.**

There seems to be a widely accepted concept that expanded memory can always increase the potential of the system to do more work, even though the memory may be asynchronous with the CPU cycles. This is false as can be seen from below and has lead to some very expensive mistakes. The severity of the problem is related to the difference in cycle time of the memory and the cycle time of the CPU. A typical example of this is found in the IBM LCS on a IBM 360/65. The cycle time for the LCS is 8 microseconds compared to 1 microsecond for the CPU. Asynchronous devices that must work together, must get in synchronization

with each other before anyting can be done. This means the faster device, usually the CPU, must wait for the slower device, in this case, the memory. Cache memory helps relieve some of the delay and speeds up memory I/O processing. In order to achieve synchronization, some form of "hand shaking" is necessary to allow each device to communicate efficiently and correctly with another device and assures that both are ready to communicate. In the IBM 360/65 an inhibit pulse can be easily monitored by a hardware monitor. This pulse from the LCS travels to the CPU and stops the actual execution of instructions. However, "the meter continues to run", the wait light stays out, accounting time continues, but no instructions can be executed. In this system, the inhibit time can easily rise to 30% of total elapsed time. In essence, the CPU is halted 30% of the time.

Today's computer systems are a collection of asychronous components including memory, CPU and channels. The fastest of these must always wait for the slower one to finish. Devices that are most closely matched in cycle speed to the CPU are more likely to have less negative impact from the inhibit. Therefore, the intuitive conclusion of adding more memory to improve performance is not necessarily true, and once again, careful measurements should be made to examine the amount of time spent in the inhibit state.

5. **Fallacy: Economies of scale or "Grosch's law" still apply in computing.**

This intuitive comment may have been more popular prior to the advent of minicomputers. Today, this may not be such a wide spread feeling since costs of hardware have dropped tremendously with the additional advantage of increased capability (CPU speed, amount of main memory, etc.). Herb Grosch, of IBM and past president of ACM, inferred that the capacity or power of a computer increased as the square of

the cost of the computer. A study referred to in Carlson [Ref. 17], indicated that minicomputers can be up to 300 times more cost effective (cost/million matrix multiplication instructions: high of $5,102 for IBM 360/30 and low of $16 for Data General Eclipse from) than the older mainframes or maxi computers.

6. **Fallacy: In virtual storage systems you can get by with less real memory than before.**

Although virtual memory does simplify many problems associated with multi-task systems, it has not been clearly established that less memory is required. In a logical sense, the user can be lead to believe that they need less real memory in a virtual system. Although the evidence is not totally conclusive, the conclusion reached in the August and September 1974 issues of EDP Performance Review on VS performance tend to indicate that additional real memory is actually required in order to maintain the pre-virtual memory system performance level.

B. SOME POSSIBLE FALLACIES

A list of possible fallacies include:

1. **Possible fallacy: The use of double or multiple buffering or a multiprogrammed computer is better than single buffering.**

According to Carlson, these does not seem to be a clear one-to-one relation between additional memory and channel tie up in terms of what's best for the total system throughput. If the data flow is in short bursts on a multiprocessing system then the system can often switch with relatively little overhead to other processing while the transfer of data is going on. Adequate data to prove or disprove this intuition is obtainable through the use of benchmark program runs with various buffer sizes, combined

with careful monitoring of the total system activity and the total job throughput.

2. **Possible Fallacy: More of a primary resource (main memory, disks, tape drives, channels, etc.) will speed up a system.**

Very often it is assumed that "more" of certain resources will improve system performance. We have seen earlier, this is not necessarily true for asynchronous main memory.

3. **Possible Fallacy: If no one is complaining, all is well.**

It is rare that people stay satisfied about something for very long. At times, programmers and users may "accept" things with the attitude that its' the best the system can do, even though, say, the response time desired is not currently being supplied by the system. Users and programmers should be encouraged to freely provide and make suggestions and make known their concerns and comments about system performance at all times.

4. **Possible Fallacy: Assembly language is faster and costs less then high level languages like COBOL.**

Assembly language could be faster (in execution) in special cases and when used in the critical secton of a program or routine. In general though, medium to large programs written in assembly language are not "faster" and less expensive then a high level language when the following considerations are involved:

a) Coding time.
b) Human debug time.
c) Machine debug time.
d) Compiler/assembly time.
e) Execution time; and frequency of execution.
f) Program maintenance time and cost.

C. GENERAL FALLACIES

A wide range of cases are covered by the following general fallacies:

1. Fallacies of generalization.

This fallacy is developed when using statistical sampling and measuring one event then generalializinge about apparently related events as well. One example is the use of CPU busy measurments to make inferences about other parts of the system or the system in general.

2. The fallacy of over-reducing complexity to simplicity.

This is inversely related to the previous possible fallacy. If a complex problem is overly reduced to a simple one, then the information obtained or the solution found may only apply to the simple problem and in no way be relevant to the complex problem. Although at times very desirable, simplification of problems must be done very carefully and not be done entirely to reduce the detailed level of data measurement and collection.

3. The Fallacy of composition.

This is the assumption that because one item of a group has a particular property, the whole group has the same property. This can occur, for example, if only one of many terminals or disk units is measured and it is assumed that all others in the group behave in the same way. Again, the data collected can usually only be used to describe the behavior of the specific device that produced the data.

4. The Fallacy of appeal to authority.

Most people are exposed to this fallacy in early life and live with it throughout their lives. Always check out the validity and extent to which someone is an "authority". If a vendor or product salesmen makes a claim, or statement, or suggests a solution to a problem, request the grounds

upon which the statement was made. Check it out by contacting other technical people that are directly involved with the product or method. For example, a system programmer may suggest that the system needs more main memory so that programs may not have to be "overlayed" in order to execute. This may help the programmers or ease their problem, but could have no, or even a negative effect on the system in general. Appeal to authority is one of the "cheapest", quickest, and easiest methods to solve a problem if the information given is correct, but it can be the most dangerous way, if the information is incorrect. Therefore it could prove to be the most expensive solution.

The next section discusses a suggested performance imporvement procedure that will hopefully help avoid the previously mentioned fallacies through the use of hypothesis formulation and verifiction.

D. A SYSTEM PERFORMANCE IMPROVEMENT PROCEDURE

Bell [Ref. 3], in contrast to figure 2.1 "A common approach to performance efforts", suggests a seven phase testing procedure depicted in Figure 6.1 and listed below.

1. **Phase-1:** Understand the System. This is part of the CPM effort in terms of management organization of the installation, description and characteristics of the workload on the system, descriptions of the hardware configurations and software programs, etc.
2. **Phase-2:** Analyze Operations. This phase also involves CPM and collects more detailed information in order to assess where bottlenecks may exist and how well the overall system is "running".
3. **Phase-3:** Formulate Performance Improvement Hypotheses. Formulate specific performance-oriented hypotheses about causes and solutions to problems and bottlenecks uncovered in Phase 2.

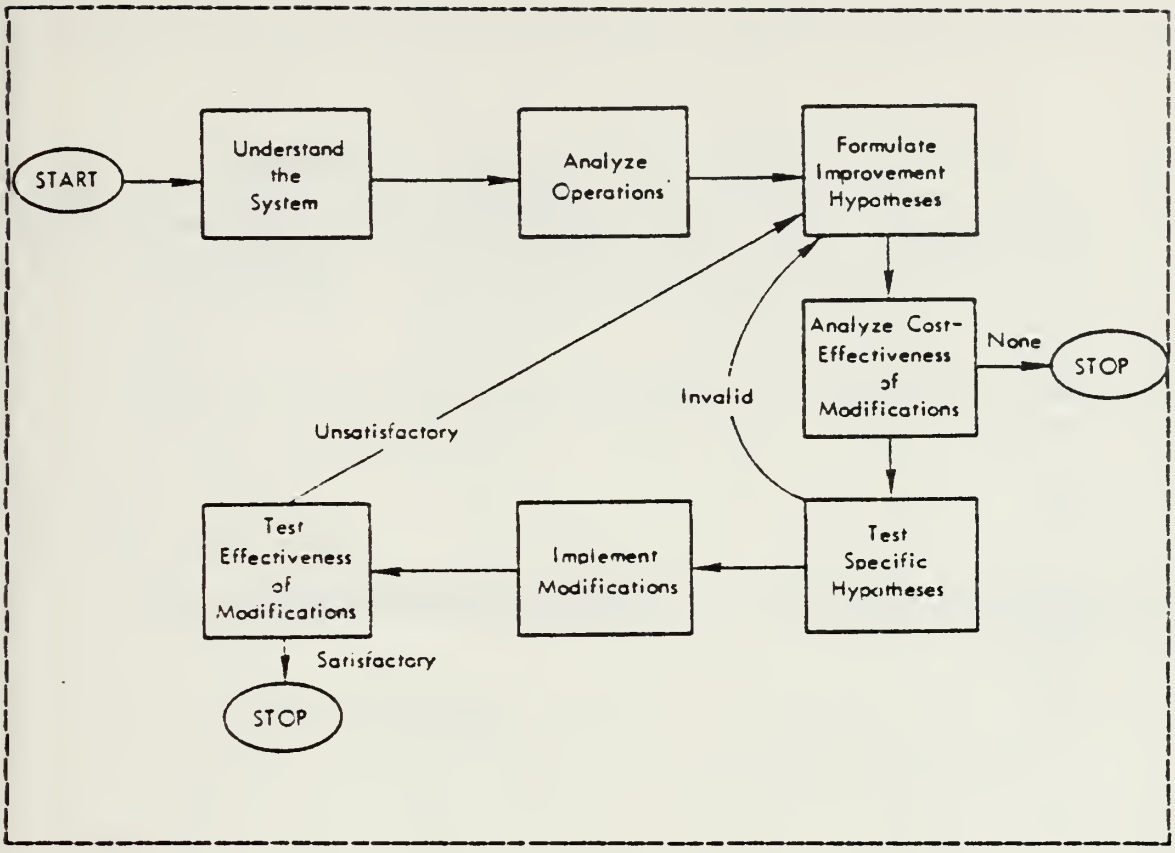


Figure 6.1 Recommended Performance Improvement Procedure.

4. Phase-4: Analyze Probable Cost-Effectiveness of Improvement Modifications. Try to establish if the result will justify the investment required by the effort.
5. Phase-5: Test Specific Hypothesis. This phase usually involves the largest effort in the CPE process. Figure 6.2 from Bell [Ref. 3], provides an orderly sequence of steps relevant to validate the hypothesis.
6. Phase-6: Implement Appropriate Combinations of Modifications. It may be worth while to consider performing several modifications at once as opposed to doing them one at a time in order to save "down

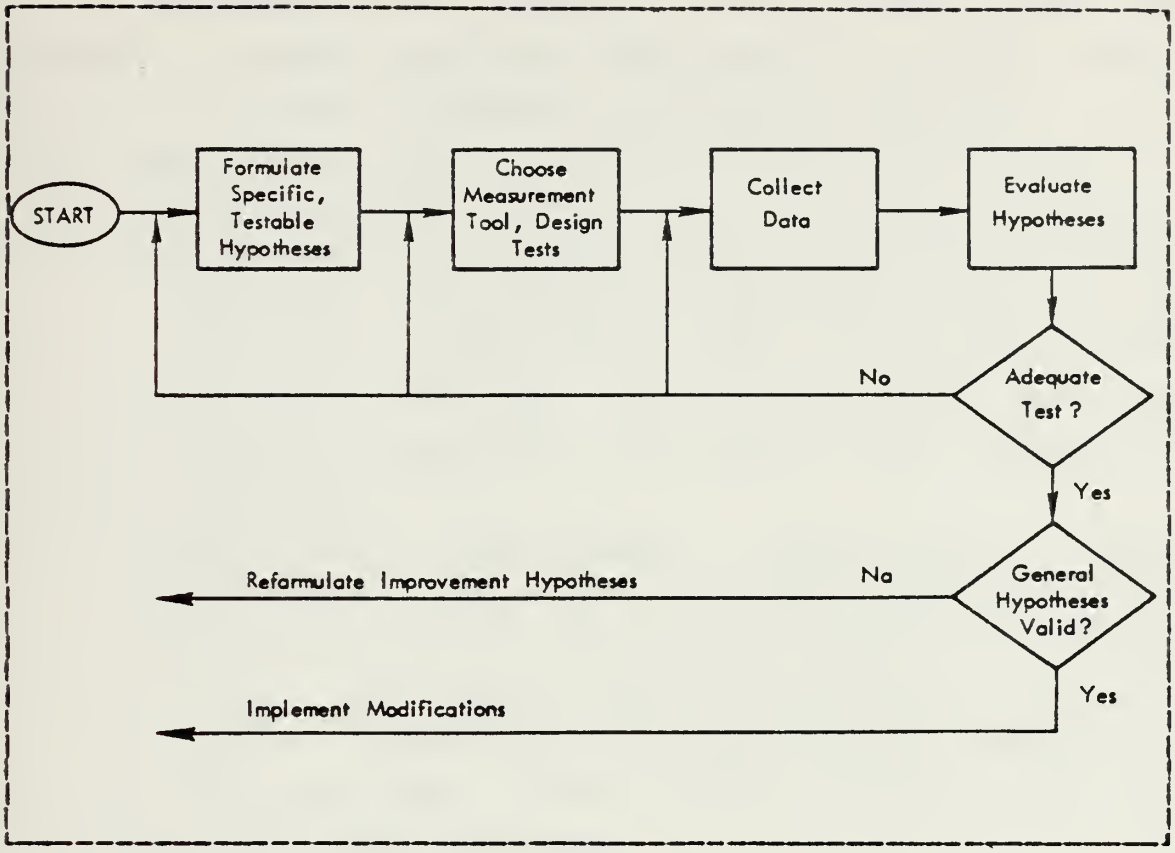


Figure 6.2 Suggested Hypothesis Testing Procedure.

time" or overhead generated by making a single modification. Analyze what impact a composite modification may have on the system and its users. Will programs have to be modified and recompiled?

7. Phase-7: Test Effectiveness of Modifications-Collect data to establish the effect the modification had on the performance of the system. If appropriate go to phase-3.

Bell provides valuable insite that is still relevent and applicable to day in the following areas: workload, operational characteristics, and system characteristics. In each area he lists several questions that should be asked.

In contrast to the fallacies above, Bell lists and describes several hypotheses relating to the three basic methods he proposes for system performance improvement. The three basic methods along with some of the hypotheses are:

1. **Reduce Workload**
   a) **Hypothesis: A Free-Good Approach Has Lead to Large Demands.** This basically means that if use of the computer is free then users will produce many programs that will take advantage of this and possibly overload the system with unnecessary workload.
   b) **Hypothesis: Unconverted Workload Has Caused Inefficient System Use.** System upgrades usually involve new and more efficient ways to do things (I/O, new language features such as FORTRAN 77, etc.). However, many programs are not modified to take advantage of new features and continue old inefficient practices.
2. **Tuning the System**
   a) **Hypothesis: I/O Contention for a Specific Device Slows Processing.** This can be a result of placing too much data or too many conflicting programs on one disk pack or channel.
3. **Upgrading the Computer System**
   a) **Hypothesis: The Computer System Should Be Upgraded to a Multiprocessor.** Increasing the CPU power of a machine may help in the short run by adding another CPU. If other perpherial devices are adequate for multiple CPUs, this can be a cost effective move.
   b) **Hypothesis: The Current System Must Be Replaced With a Larger System.** This can result from the workload outgrowing several components of the system not just the CPU as in the previous

hypothesis. Upgrading within the same computer "family" helps to reduce conversion costs and may be the most cost-effective. However, caution must be taken and a review of alternative machines should be considered as well. A trade-off of easy conversion for a modest in stead of major increase in CPU speed may not be justified.

E. SOME POSITIVE GUIDELINES

The following axioms found in [Ref. 17], were adapted from Fisher's "Historical Fallacies" [Ref. 18].

1. Questions relating to performance issues must be operational. That is, they can be answered with factual evidence (supportable by measurements).
2. The questions should be open-ended to allow exploration, but they should not be so wide open that there is no direction.
3. The questions should be flexible so we can adapt them to new information. There should be no early hardening of the categories.
4. The questions must be analytical. This means they can be broken down into smaller parts so we can obtain answers to manageable pieces.
5. The questions must be both explicit and precise so that others can truly understand what we're trying to get at and can concur with the answers found.
6. The questions must be tested against actual operating systems.

Common sense is a good tool to use. However, as previously shown it can easily lead to fallacies of intuition. To use this tool alone would be foolish and possibly wasteful. Therefore, common sense should be used to develop a hypothesis, backed up by precise conclusive measurements.

With each collection of performance data, a simple binary choice is not usually avaliable. Instead of just one fork in the road there can be an overwhelming number of dead-end choices-easy to get on and follow, but headed in the wrong direction. Performance evaluation is an extremely difficult task, and can be a wasteful use of valuable resources if done improperly. Experience is the best compliment to common sense. Both work together to increase the ability of the individual or group envolved in system performance improvements. With addition of meaningful measurements to validate or invalidate a hypothesis or intuition, system performance can be realized.

VII. IMPROVING PERFORMANCE WITHOUT MONITORS

"Improving the performance of an application system doesn't always require hardware or software monitors, staffs of specialists, nor complicated timing diagrams. Some of the best results may come from just looking at the code.", [Ref. 11]. Computer programs are usually not written under the most ideal circumstances by the most ideal programmers. Many programmers don't become too concerned with the impact the program they are writing, or more correctly-their "style" of programming, will have on the performance of the system upon which the program will execute. This view is supported by subtle and not so subtle evidence commonly found in programs. For example, many programmers will needlessly initialize an array to zero, even though they will completely fill the array with data. A less obvious example is the choice of data type (real vs integer) without reguard to the number of CPU cycles required for the alternative choices. Therefore, even the most efficient configurations of hardware and system software can still produce unsatisfactory performance because the applications that are executed on it may not make the most effective use of the system resources.

The following circumstances can have direct and/or indirect adverse effects on the performance characteristics of an application program or systems program [Ref. 11] :

- Competence of the program designer.
- The mandate given the designer.
- The time frame under which the "program" was produced.

- The reward system in effect during the writing of the program.
- Conversions from one hardware or software (operating) system to another.
- The number of modifications made to the system.
- The existance and enforcement of definitive programing standards and quality control.

The list of circumstances that can effect program performance can become very long. Therefore, the chances of improving overall system performance through the improvement in individual program performance has great potential. Jalics [Ref. 11] believes that, "Sometimes systems can be made to operate two, three, or four times as fast with little effort". He states further, "the task of performing such measurement, evaluation, and enhancement is not very complex and does not require special skills".

Jaclis also feels that there are two basic principles to be used in performance measurement:

1. "Each system has a small number of critical parts (these critical parts system[2] typically comprise 1% to 10% of all program statements in a system)."
2. "The performance of the entire system depends to a large extent on the performance of its critical parts (that is, one can just about ignore the non-critical part)"

This strategy for making the more efficient is quite simple and consists of finding all the critical parts of a system, and modifying the areas that will have a significant impact on the positive performance of the system. Jalics feels

---

[2]In the context of this chapter the word "system" is defined as a collection of jobs. Whereas jobs are defined as a collection of one or more programs. A job step is one program.

that this same strategy can be applied to the whole set of application programs. What needs to be done therefore, is to concentrate on a very small part of the whole (less then 10%) system. Desirable selection criteria, according to Jalics are those few application systems that:

1. Are time-critical, that is, systems that require results a very short time after input data are supplied.
2. Are run frequently and consume considerable system resources, or
3. Consume extremely large amounts of computer resources, even if they are not run very frequently.

A. AN APPROACH TO FINDING CRITICAL PARTS

The best source for the determination of what jobs constitute the ciritical section, using the above selection criteria is daily and monthly job report summary reports. The task then, is to isolate those programs that make up the critical part of the system, find those parts of these programs that are executed most often and review other characteristics of the program that may contribute to it's inefficiency. Jalics provides some guidelines to help accomplish this.

1. Check Out the Compiler

Most programs written in high level languages are feed as input into a program called a compiler. Most compilers are out of the range of modification by most programmers unless they have had compiler design and writing experience. However, programmers can at least learn and become aware of the performance or efficiency characteristics of the compiler. Specifically, data types and structures and certain language features can be investigated as

to their degrees of efficiency. "Data type implementation alone can vary by a factor of 2 to 40 (one can be 40 times slower than another)" [Ref. 11].

A simple method by which to look at compiler characteristics would be to write a program that contains all the data types and language features of interest. After compiling the program, an assembly listing of that program can usually be obtained. This listing can provide information about the number of assembly instructions executed per high level program statement. For library routine calls, a machine instruction trace can be used, if avaliable.

Experiments were run on some of the features of a COBAL compiler residing on a UNIVAC 1100. Although the experiments were run against one specific language, it would not be difficult to apply and conduct similar experiments on other languages. The following were among the fifty-four tests made by Jalics :

1. Arithmetic statements with various combinations of operands--different data field sizes, various alignments.
2. Movement (I/O) of data fields with sources and destinations of various sizes and alignments.
3. IF conditions using various lengths of items, alignments, and use of condition-names.
4. Table searching using loop, and search verbs, using various structure tables, indices, data items, and alignments.
5. Language features that examine fields using various lenght fields and alignments.

The following conlusions were drawn form the results of the experiments:

1. Arithmetic operations can differ in execution by as much as 30 to 40 times depending upon data item type.

2. The most efficient data field sizes may be a multiple of bytes and or words. The use of these efficient field sizes may increase processing 2 to 3 times faster.
3. Alignment plays a tremendous role in efficiency of both arithmetic and character manipulation. Desirable data fields are single characters (no alignment required), three characters (half-word aligned), multiples of six characters (full-word aligned).
4. Avoid testing large data fields whenever possible. The use of condition names for more complicated testing offers no advantages.
5. Built in search verbs are not always more efficient than a program loop depending upon the index used. Alignment of each table entry to a word boundary provides significant performance improvement.

The input/output facilities are one of the most important efficiency aspects of a compiler. Efficient record and block sizes are a function of the characteristics of the physical storage devices, as well as the size of blocks most effieiently handled by the operating system. The size of blocks can range from several hundred to several thousand bytes.

2. Efficiency Measurement

An important first step before any performance enhancent is to get a measure of current efficiency. What is needed is a way to relate the amount of useful work performed to the computer resources it took to process the job, in order to come up with a measure of work that is easily understood by the users of the system and the operations group. Examples of measure of work may be the following: a schools registar's office may have a program

that does a certain amount of processing in relation to the number of students enrolled in the current academic quarter (i.e. "X" number of student records per system unit of time), or a bank may do a certain amount of processing for each tranaction that day ("X" number of transactions per system unit of time)

Having obtained a measure of work, an Efficiency rate (ER) in terms of Units-of-Work (UOW) divided by Computer-Resource-Units (CRU) can be defined, [Ref. 11]. In the case of a UNIVAC 1100, the job accounting system provided a resource unit measure as a weighted sum of CPU seconds and I/O seconds CRU's can then be thought of as system-seconds with one CRU equal to the complete use of the computer for one second. Then the ER of the registar's record keeping program may be 54 student records per CRU, and the bank's ER may be 310 transactions per system-second.

The next step is to compute the ER for every run of a job. This makes it possible to compare system efficiency before and after performance enhancements are made, even though different data is used in each run.

3. Obtaining the Efficiency Units

It is highly desirable to compute and make available the ER at the end of the execution of a job. This information can be placed in a file for review by interested personnel throughout the day and then tabulated into a daily ER/job report or summary.

If the job consists of only one program, calculating the ER is straight forward, since the UOW and the CRU are easily obtained. This task becomes somewhat more involved with jobs containing multiple programs in that UOW and CRU data will have to be collected and summed at the end of the job.

4. Finding Critical Job Steps

If multiple job steps are envolved, computer usage should be captured at the end of each job step or program. Two sources of this information are: the operating system may automatically put this information in a log or there may exist job control language commands or supervisory calls that will make avaliable the CRU information about the job step. Having this information about all programs that make up a job helps to identify the critical job steps or programs.

5. Critical Job Step Selection

Having run the system several times, and identified the critical job steps, a similar procedure is used to locate the critical sections within a job step or program. Once identified, the critical sections of a job step should be looked at using a compilation listing of the program with a cross-reference index of all data names.

6. Tune the Program and Compare the Results

The next section gives specific information on tuning a program. The critical section should be analyzed for inefficient code or practices and modifications suggested to correct these. After having implemented all modifications to the critical section of a job, using tuning techniques similar to those in the next section, the above review process should be started anew. Different areas may now become critical job steps as a result of modifications to the "old" crtical section. This process should be repeated as long as it felt that the rusults justify the efforts, which is a decision unique to every situation. Guidelines on making this decision should be incorporated in the CPM plan of the facility, as well as in the individual CPE effort.

B. PROGRAM TUNING HINTS

Program tuning is within the capability of most programmers once they have an idea of what it is about. As mentioned earlier, there is potential for significant program performance improvement, that in turn can lead to overall computer performance improvement.

Jalics [Ref. 11]. provides list of do's and don'ts along with ideas related to program tuning practices. The following were obtained from this list:

1. **Find the innermost loops in the program.** A visibly well constructed program may give some hint as to what code may be executed most often. Since the contents of a data variable may determine the activity of a loop, programmers could write a program using hooks to identify those parts of the job step or program that are most active. These parts usually involve loops or innerloops.

2. **Next, move all superfluous statements outside of the critical section.** Such statements include initializing constants, initializing records to zero or blanks even though they will be filled with new data each time, initializing whole arrays to zero when not necessary, etc.

3. **Try to locate inefficient uses of data types and convert them to the most effiecient type within the intended precision, etc.**

4. **Investigate data field sizes.** Different computers may have different optimum size boundaries such as quarter word, halfword or full word. For character data, it may be more efficient in terms of processing overhead, to use storage fields that are multiples of words to avoid alignment overhead associated with partical word fields. The overhead associated with

alignment beccmes much more obvious when inside a loop that may have several thousand iterations.

5. In complex conditional statements, test for what is most probable first. In the case of an IF statement with multiple conditions, place those conditions first that are likely to decide the result of the ccmpcund condition without testing all the conditions. When conditions are ORed, the condition most likely to be true should be checked first, for ANDing, the condition most likely to be false should be checked first. The "likelihood" information required in the above conditionals is not always known, but when it is know it may prove very worth while to make an effort to take advantage of it.
6. In a situation involving a series of consecutive IF statements, the IF statement most likely to be true should come first then the next most likely second, etc.
7. Avoid testing large data fields. For example, it may be necessary to load a large data field with all zeros or blanks but dosn't directly check the whole field to see if it is indeed all zeros or all blanks. Rather, use a "flag" (a small status field) or set a bit in the field indicating the fields zero or nonzero status. Checking a few bits over several words may save considerable time. This method should be used for any large data structure when appropriate.
8. If possible avoid the use of verbs that are very time-consuming such as COBOL's TRANSFORM, INSPECT and EXAMINE whenever possible. One example inJalics [Ref. 11], used EXAMINE on a record with 932 characters which resulted in 6,127 instructions being executed.

9. **Beware of language interfaces and subroutine call overhead.** Overhead associated with certain language invoked constructs can vary significantly. In COBOL for example the efficiency of the PERFORM verb versus the CALL verb should vary greately depending upon whether internal subroutines are used or external subroutines are used. Upon every subroutine call, some compilers will generate code that will dynamically obtain more memory to save the environment of the calling routine. Upon subprogram termination, the memory is then released. According to Jalics some compilers have little known options that will either acquire additional memory dynamically or else reuse it each time the subprogram is called. A simple example which can be very inefficient in terms of time and space involves calling a subroutine to compute a value based upon the contents of a field in a record. Many consecutive records processed may have the same contents in the field and a subroutine is called each time to compute the same value. It would be much more efficient to retain the contents of the field and computed value of the last record processed and compare it with the current record and only compute a new value by calling the subroutine when they differ.
10. **Know the options available on the compiler.** Some compilers have options that provide several levels of effiency. Some compilers have memory efficient or execute time efficient options. Code that executes inline may take up more memory but execute faster then code placed in routines and called, which uses less memory. Many other options are available that may range check data structures and subscripts, place unnecessary code on the outside of loops, etc.

However, some of these options are intended to be used only during the final compile, and if turned on during program development can waste significant amounts of compile time. Therefore, constant use of inappropriate compiler options during all phases of program develcpment may have an adverse impact on overall system performance.

11. **Take a close look at tables or arrays.** Certain alignment characteristics may make the compiler generate efficient code for arrays and tables. This is true in machines that are word oriented and operate much more efficiently on tables where each entry is word aligned. This is a trade off between time and space, since table elements may have to be padded to take advantage of the alignment efficiency.
12. **Look at the subscripts used for addressing tables.** In COBOL for instance, subscripting can be more efficient than INDEXING.
13. **Look at table search techniques.** In certain cases indexing (ie. storing xxx in location xxx) can be much more efficient then simply searching a table [Ref. 11]. The indexing method is very direct. If searching is required make sure the most efficent method is used. This may depend upon the order or arrangement of the data within the data structure. In some cases, it may be more efficient to order the data to achieve better search efficieny.
14. **Is the current read requesting the same data as the last read?** It may be worthwhile to check the current read request to see if it is requesting the same information as the last read. Some applications may have a high cccurence of this, in which case many unnecessary I/O transactions can be avoided.

15. **Review the processing technique of a program,** is it the most efficient? Random access techniques allow the processing of records in any order. However, as mentioned earlier, ordering the data in the sequence in which it will be processed could cut down on disk access overhead time in large files, since the data would hopefully be stored to provide consecutive, inline disk accesses.
16. **Is file organization optimal?** Three commonly known file organizations are sequential, indexed-sequential, and direct. The choice of organization used for a file for a particular application can substantially impact performance of the whole system.
17. **Look at file blocking buffer size.** The programmer that is aware of the size of physical blocks which are efficiently processed by the operating system, the access method, and the various storage devices will be better able to produce efficient I/O. Programmers that are unaware of these are more apt to write inefficient code and negatively impact performance. Block sizes on a disk usually correspond to a sector and the block size for an operating system is constrained by how much memory was allocated for the I/O buffer.

VIII. CURRENT AND FUTURE TECHNOLOGY AND PERFORMANCE

Increasing end-user computer power, micro computers, networks, data base machines and database management systems, multiprocessor chips, multiprocessor computers and fifth generation computers are all relatively new fields and technologies that will most likely demand some type of performance evaluation. This chapter will explore some of these technologies in terms of performance evaluation. New performance parameters will have to be defined as well as tools and methods to measure them. The research literature avaliable in this area is limited.

A. END-USER COMPUTER POWER

As a result of the favorable economic trend in the computer hardware industry, complete computers with respectible computer power can reside on less than one half of the typical office desk. Performance issues in this area concern the capacity and reliability of the communication linkage between the main computer system and the user equipment; the adaptability of this end-user equipment to the various experience levels of the group of users; and the amount of conversion of non-standard equipment and software required in order to be compatable with the main computer system.

B. MICROCOMPUTER PERFORMANCE

It is not long after acquisition that the microcomputer user realizes the potential of sending or receiving information or data to or from other microcomputer users. This involves communication lines (usually existing telephone

lines), modems and compatability-the ability of the computers to understand what each is sending to the other. If we consider the interaction of the user with the microcomputer as being part of the overall system, then user-machine and user-scftware interface "friendliness" can impact performance of the system.

1. Communication

As discussed in an earlier chapter, the rate at which data or information is passed through a modem can severely impact the performance of the computer. Use of a slow, 300 baud modem between two systems can produce the same performance as if poor terminal response time and system slow down due to a heavy workload. The solution to this problem is simple but expensive. The user should purchase the fastest modem affordable. Unfortunately, the price of 1200 baud modems is considerably more than the slower 300 baud modems, and most home microcomputer users have 300 baud modems.

2. Compatability

The other performance issue requarding microcomputers that communicate with each other is compatability. If two microcomputers are incompatable and can't communicate directly with each other, then conversion or interface programs/hardware are needed. This adds overhead and reduces the performance of each computer since time must be spent converting the data or information to a format which is understandable by computer. A standard interface chip placed in each communicating computer could be a solution to this problem. The issue of incompatability becomes a night-mare when a large organization like an insurance company allows uncontrolled proliferation of microcomputers and then decides to tie all departments together into a computer network within the organization.

3. User Interface

The microcomputer has exposed the computer to masses of users. Many of these users simply can not type and are frustrated by the use of keyboards that require some degree of finger dexterity. Many interactive programs and packages require typed responses, which result in reduced system performance when used by this class of users. Therefore, alternative devices are required to help improve the interaction of the user with the computer. These devices do exist and are called "picks" and include the mouse, the light pen, the joy stick and the touch panel. Touch panels seem to have the most reliability since they involve the users finger or a pencil and a grid system that detects the location of the pointing device relative to its position on the screen. Therefore, the user simply points to an item in a menue that provides the user with a choice of responses. Newer systems use touch panels together with icons-words or ideas represented by an image.

4. Inexpensive Software

Since so many microcomputers have been sold, the demand for software, expecially inexpensive software is great. This tends to produce both application and system software that may not be performance conscious. Cheap software may run but it may run very slow and use excessive main memory. Therefore, software should not be purchased by price alone but by its reputation if possible.

C. NETWORKS

Telecommunications and networks are becoming very popular and should see wide spread use in the 1980's. Designers of these technologies are providing a great varity of choices in the area of data carriers and data terminators:

- Front-end processors
- PBX (private branch exchange)
- Modems
- Voice satellite links

Performance evolution in this area has been hampered by almost daily changes in communication technology.

Network design, development and testing tools cover a broad range of analog, digital, and protocol analyses. The physical components that these areas deal with are: carrier lines, modems, and other forms of communication. Because of the potential size of networks (several thousand nodes), only software tools can cover the overall global nature of the network. These software tools help network designers plan, analyze and optimize the performance and cost of the system. Analysis program software gathers message traffic information, measures message response times, determines the probability of blocked messages, and ultimately defines the maximum traffic volume the system can carry within acceptable response times.

Test objectives of protocols include overall evaluation of the software and control sequences from one end of the system to the other end of the system. On a character-by-character basis, timing signals and properly controlled synchronization are reviewed as well as error detection in the following three categories: characters, coding, and parity.

A central data base is created and used at the system level design of a network. Various programs, (using parameter information from the data base), model many possible configurations and analyze them in terms of their performance and cost. The sheer complexity of data communciation networks (some may eventually have 7000 on-line

units) necessitates software monitors. This softwareis supplied by the vendors of front-end processors (e.g., the IBM 3705 communicaticns processor). The programs analyze the data traffic controlled by the front-end processor and measure such information as the number of busy signals and the number of nonresponding units. The advantage of vendor supplied programs is that they have are able to access all communication lines controlled by the processor. In contrast, protocol analyzers and other test hardware examine only one line at a time.

Network performance can be hampered by failed stations or transmission components. Failure analysis software examines the avaliable channels and the expected number of failed devices per operating period. This is done by calculating the effectiveness of the fault-diagnostic, fault-control, and outage-prevention measures that are built into the system. Expected transmission errors in the connection links themselves are studied and added to the expectation of overall system performance.

By examining the potential data traffic capabilities and subtracting the effects of blocked messages, faults, and so forth, the designer can measure the performance of the system with regard to thoughput (bits or characters per second), messages per interval, and message response. Given information on traffic variations, the system's sensitivity to volume can be projected on an hourly, daily, and annual basis. If a given configuration meets the raw thoughput, traffic volume sensitivity, and reliability factors, its construction can be optimized to lower costs.

All of the system's performance measurements can not be optimized simultaneously. It is in these instances that performance analysis software proves its worth. The programs indicate the tradeoffs among various components and methods, which allows cost-effective system to be designed

with a reasonable response time and with minimum performance degradation in the face of violations in traffic, error rate, or failure rate.

Data communication tests includeanalog tests on the carrier line and digital tests on the interface between modems and data terminal equipment (computers and terminals). The protocol-level tests evaluate data and control sequences character by character. See figure 8.1 for a graphical representation of the data communications testing environment.

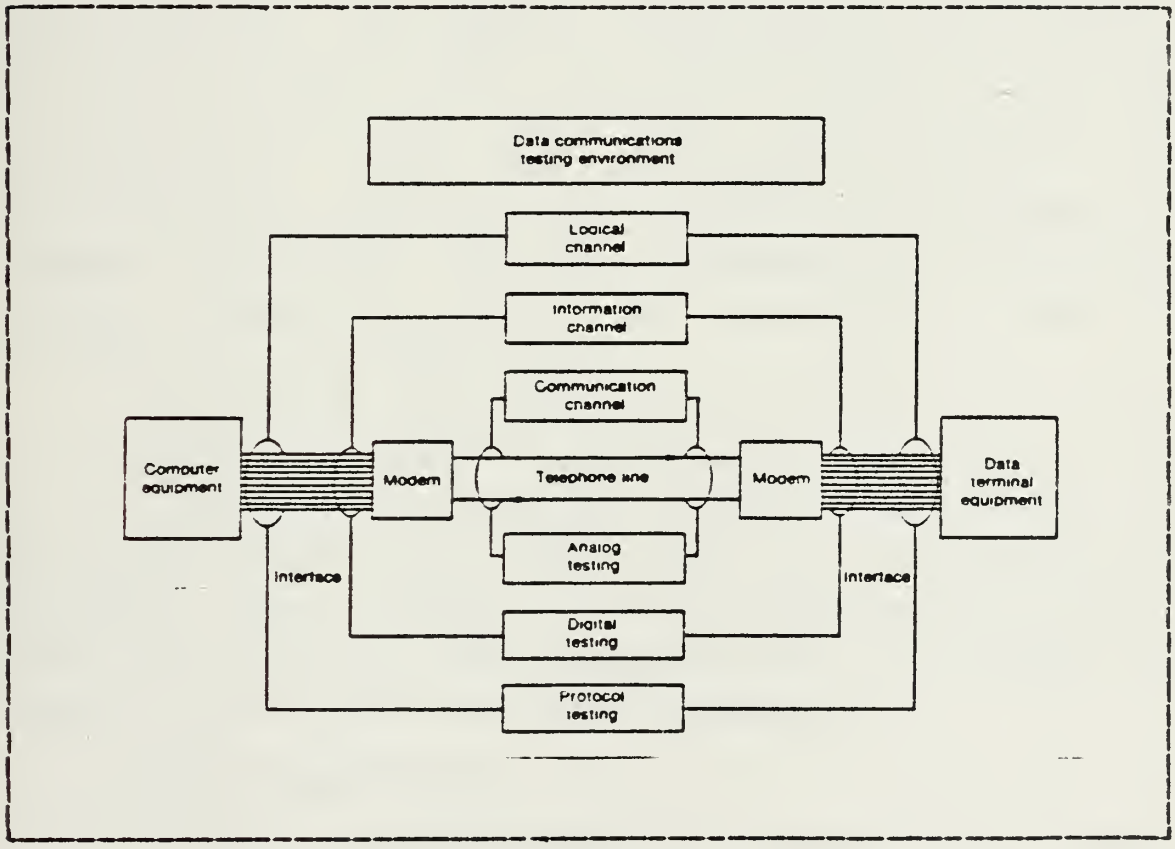


Figure 8.1 Data Communications Testing Environment.

1. Communication Line Varification

Existing telephone lines are used by most networks due to the low cost and wide spread availability. However, these lines are not designed for the high speed, error free transmissions desired by the network and therefore cause problems. Factors that can affect the performance of a communications line are analog parameters covering bandwidth, transmission mode, and transmission problems. Therefore, the analog problems reduces itself to testing individual lines and determining if the line meets the network requirements and specifications. TIMS (Transmission Impariment Sets) can provide comprehensive tests and measurements of an analog line. Measurements attainable by this device are noise, the gain slope, noise-to-ground and signal-to-noise ratios, and peak-to-average ratio. The peak-to-average ratio is a quick benchmark measurement and "gagues the overall spreading or smearing in time of a signal transmitted across the communication channel" according to Bailey [Ref. 19].

Measurement and testing of the carrier line is best accomplished end-to-end by tramsmitting a test signal at one end to a meter at the other end to receive and measure it. This simple method will indicate which end, transmitting or recieving, has a problem. If the communications line is found to be reliable, then the problem must be in either the modem or the DTE (data terminal equipment).

2. Data Verification

Digital testing is required to verify the data being transmitted and received over the lines. By sending a fixed pattern test packet through the system and using a data simulator or protocol analyzer in place of certain system components, data transmission errors can be identified. If

the data becomes suspect, the modem can be checked with a pair of BERT's (Bit Error Rate Testors). One BERT sends a data sequence to the transmitting modem and another BERT analyzes the data as it passes through the receiving modem. A bit error ratio can be established and is a good measure of overall link performance. A typical voice communications link should have a ratio not exceeding one error in fifty thousand bits, according to Bailey. Measurements capability of BERT's vary with manufacturer and include parity check, block error rates, carrier loss, clock slips, and timing distortions. The distribution of errors over a given bit stream is called block error rate. Throughput of a system that is capable of block-error detection recognition is defined as the ratio of error-free blocks to total blocks transmitted. Blocks that are found to contain errors must be retransmitted. This impacts throughput rate, which in turn impacts performance of the system. A large bit error rate in conjunction with a low block error rate would provide better performance than an equivalent bit error rate distributed evenly across the data stream.

3. General Performance Summary

Problems such as carrier loss, clock slippage, and timing distortions warn the designer that problems exist within the analog carrier line. The proper operation of modems and carrier lines can be verified, with the aide of a good bit error rate tester, through the use of an algorithm that isolates problems to various components of the data communications link, see figure 8.2 .

The next step is testing the DTE, which uses data simulation, data monitoring, and protocol analysis. Protocol analyzers have the same capabilities as data-link monitors and simulators, as well as their own data analysis functions.

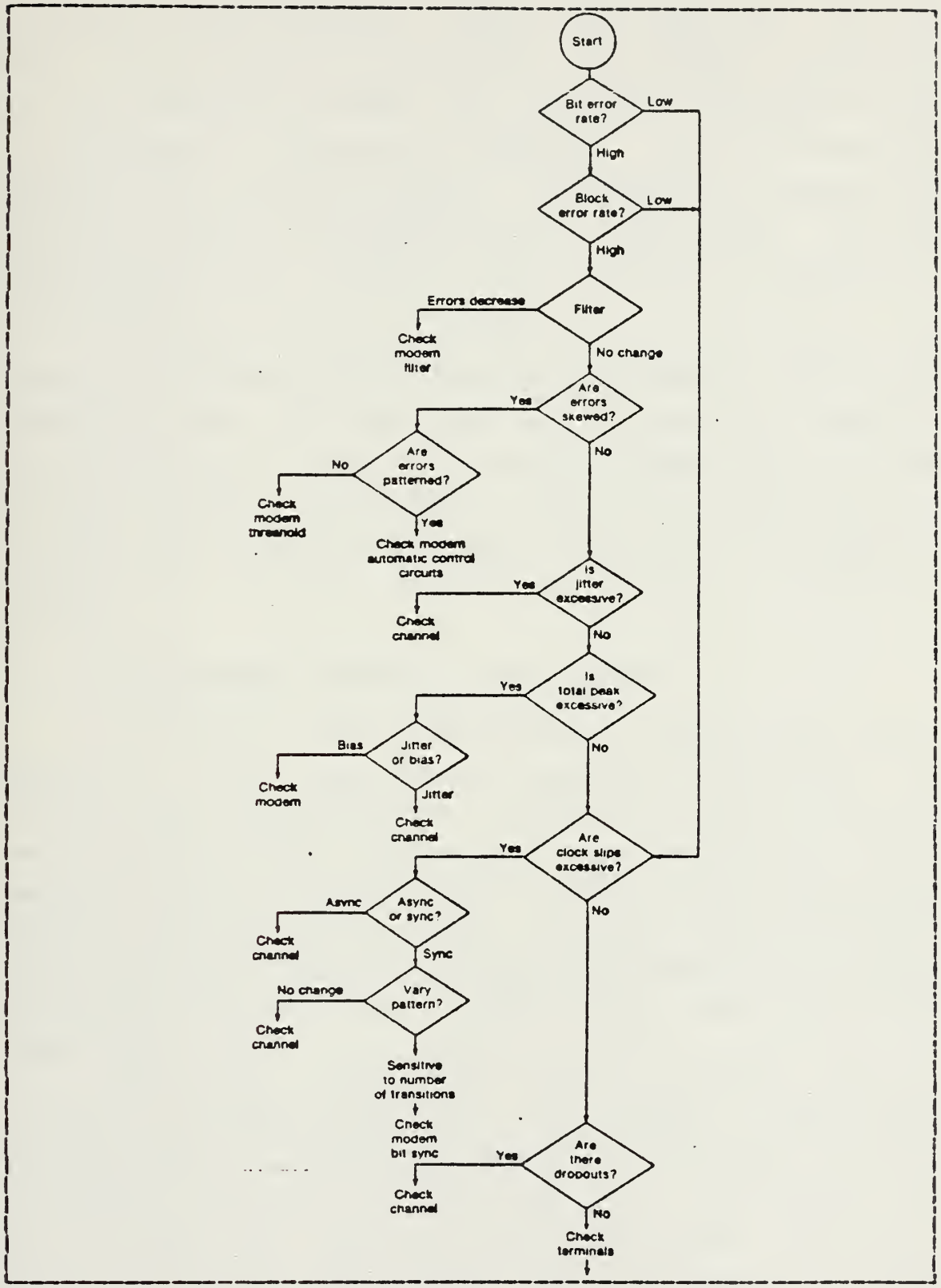


Figure 8.2 Trouble-Shooting Algorithm.

When serving as a monitor, the protocol analyzer is attached to the interface between the modem and the DTE, which may be a CPU or a terminal. No overhead is incurred by the operating equipment from the data analyzer because the analyzer is connected in parallel. Once the analyzer captures the transmitted data, the operator observes the data and protocol characteristics to find errors. He then either freezes the frames to catch the data or operates at slow speeds.

Functions of the equipment at either end of the channel are duplicated by the analyzer when it is in the simulation mode. It tests the part of the data communication link that is down or generates errors that test the ability of CPUs or terminals to recover from transmission or protocol errors. The instrument can also test new equipment before it is added to an existing network or laboratory system.

4. Performance Checking Intel Network

The NDS-II is an Ethernet-based network under development by Intel. Performance issues of the network are exaimed using hardware tools. A number of Intellec development systems are tied together by a resource manager, which handles disk storage and files, to form a resource-sharing network of development stations. A plugin controller board, that contains an Ethernet VLSI chip set and an 8- or 16-bit processor, connects each station to the Ethernet network [Ref. 19].

Data paths are analyzed by two different hardware monitors in the NDS-II system. One monitor, a NCR Comten Dynaprobe data monitor and analyzer, is capable of monitoring data lines of up to 10 MHz with 10-ns resolution. Signal levels and tranctions are counted using the instrument's 16 counter-timer probes when attached to a data

communcication channel. This monitor measures the time each I/O or disk channel spends in various activities.

The second hardware monitor is a proprietary performance analysis tool, which measures CPU activiites. It is somewhat similar to an in-circuit emulator, but only acts as a passive data monitor. It is composed of two Multibus boards and a buffer board. The processor to be tested is removed from it's slot. The buffer board is placed into the socket connectors of the processor to be tested, which is pluged into the buffer board. This establishes the analysis tool in a positon to monitor the CPU's address, data, control, and status lines and count the number of address hits. When a trigger is tripped, it can record the following information: reads and writes to memory or I/O, operational code fetches, and, when monitoring an 8086, cp code execution.

Intel, through the use of both analyzers, was able to pinpoint bottlenecks and to increase the number of supported users per link from 8 users to 16 users.

D. DATA BASE MANAGEMENT SYSTEM (DBMS) PERFORMANCE

DBMSs are experiencing a constant increase in popularity and use. A DBMS is a special purpose, complex software/hardware system, requiring dedicated resources of its own, plus substantial resources from the host computer system. DBMS performance can have a significant impact in overall host computer system performance. The design of DBMS benchmarks and the interpretation of measurement data is presented in a paper "Experiments in Benchmarking Relational Database Machines" at the Third International Workshop on Database Machines [Ref. 20].

1. Model Application

Aitken [Ref. 21], describes a DBMS model that provides performance evaluation and prediction information and can be used to:

- Predict the performance of a DBMS for a given data base design and DBMS workload.
- Investigate DBMS performance relative to changing workload, and perform DBMS stress-test studies.
- Evaluate alternative logical and physical data base designs to achieve a reasonable performance-efficient data base design.
- Support DBMS performance tuning and investigate DBMS performance problems.

The modeling approach results in a DBMS model which closely simulates the operation of a real DBMS. ECSS II (Extendable Computer System Simulator II), a special purpose language for constructing discrete-event simulation models of computer systems (and also a super-set of SIMSCRIPT) was used to implement the DBMS model.

The model uses a simulated Data Base Control System (DBCS), which handles the user interface and manages access to the data base. A simulated data base appears the same as the real data base conceptually, but contains no real data. There exists a one-to-one correspondence between the pages and records in the real and in the simulated data bases. DBCS operations include the following:

- Data Base Area Management
- Buffer Management--Strategy and number of buffers
- Data Base Space Management--page overflow
- Journaling--manages a file of before/after images of DB

- Data Dictionary Access

The model uses a Data Definition Language (DDL) to describe a complete data base schema. A Data Manipulation Language (DML) is available which includes most of the operations provided by the real DML language.

A run-unit application program is represented procedurally in the model and consists primarily of DML statements [Ref. 21]. Since the models DML statements so closely follow the actual DML statements, the modeling of an application program's data base activity can be relatively easily accomplished.

2. Model DBMS Performance Related Outputs

An automatic feature of the DBMS model collects and reports a variety of DBMS and data base related performance statistics without user intervention and are summarized below:

For each area of a modeled data base:

- queuing and utilization
- number of pages read and written
- page access time

For each simulated run-unit:

- total execution time
- number pages requested
- number pages read and written
- records stored

For each physical file:

- number of reads and writes

- number of seeks and zero-time seeks
- seek time
- data transfer time
- total access time

For the DBMS buffers:

- queuing and utilization
- number pages requested
- number of pages read
- number modified pages written

For the DBMS:

- run-unit DML request queuing
- run-unit concurrency level

The DBMS model closely resembled the actual DMBS and made it very easy to use according to Hsu. Validation of the model plus the addition of more details are currently underway.

IX. CONCLUSION

Ideally, every computer installation should use Macro Computer Performance Evaluation--CPM and be able to determine when and how to use Micro Computer Pperformance Evaluation--CPE. Ideally, CPM and CPE would work togerher to optomize the perfcrmance of the computer system. Muchsel [Ref. 22], suggests the ultimate computer performance solution:

"A good solution to this problem is, in our opinion, a program we would like to call the 'automatic operator' (AUTOP). It analyses system behavior and controls parameters to meet the operational goals of the computer installation. It either totally regulates operation of the system, or gives advice to a human operator whenever system behaviour deviates .from previously set desired values. System behavior is analyzed by a modified performance measurement program. How could one optimize system behaviour by such a procedure? The methods are to control response time; to control CPU and I/O share of certain types of jobs, possibly giving priority to some of them according to installation-specific criteria; to control system parameters according to time of day and workload; to recognize bottlenecks in the system (and eliminate them if possible); to give warning when system behaviour deteriorates; and to keep humans well informed, reporting system utilization to human operating personnel and tuning group."

Ideally, computer system designers or architects will take into account prcblems of performance measurements and their interpretation when designing new systems, and manufactures of computers will impliment their ideas as options

or standard equipment in the computer system that they make. Ideally, software designers will incorporate computer performance measurment concepts into the operatiing system and related system software. Ideally, more and more computer facilities will use a "preventative" computer performance evaluation approach that will replace the "emergency room" approach that most unprepared computer facilities are forced to use when a severe performance problem arises.

Realistically, the current trend of decreasing hardware costs and increasing hardware capability seem to have diluted the performance concerns of management in many of today's computer facilities. There may be a tendency, therefore, for computer facility management to rely solely upon the "appeal to authority", usually the vendor salesman, for a solution to performance problems instead of getting involved in a CPM and CPE effort.

It is intended that this guide provide some reasonable middle ground between the ideal situation and the realistic situation that the world of computer performance seems to be in today. In conclusion, the following steps are suggested when considering performance of a computer system:

- Develop a CPM Plan initially setting simple and realistic goals based upon the experience of the CPE personnel and include all phases of the computer life-cycle.
- Implement the best and most suitable Accounting package affordable.
- Locate "critical sections".
- Tune critical section software.
- Acquire more sophicated CPE tools only when existing tools are considered inadequate and the CPE personnel are capable and experienced enough to use them.

- If appropriate, consider outside help.

Regretfully, time did not permit the application and experimentation of some of the suggested tuning hints given in this guide.

APPENDIX A

SOURCES OF INFORMATION

The following list gives additional sources of information. EDP Performance Review annually provides an excellent annotated bibliography of the preceeding years' performance related literature. The name of the source is in bold type followed by address information.

CMG Conference Proceeding
Computer Measurement Group
P O Box 26063
Phoenix, AZ 85068

CMG Transactions
(same address as above)

Computer
IEEE Computer Society
10662 Los Vaquros Circle
Los Alamitos, CA 90720

Computer Performance
Butterworth Scientific Ltd.
P O Box 63
Westbury House Bury Street
Guilford, Surrey GU2 5BH
England

Computing Surveys
ACM
11 West 42nd Street
New York, NY 10036

CPEUG Conference Proceedings
National Bureau of Standards
Institute for Computer Sciences
and Technology
Washington, DC 20234

Datamation
Technical Publishing
1301 South Grove Ave.
Barrington, IL 60010

ECOMA Conference Proceedings
European Computer Measurement
Assoc.
Scott Yasler
Scheuchzerstrasse 5
CH 8006 Zurich
Switzerland

EDP Performance Review
Applied Computer Research
P O Box 9280
Phoenix, AZ 85068

IBM Systems Journal
IBM
Armonk, NY 10504

on International Conference on
Computer Capacity Management
Institute for Software Engineering
510 Oakmead Parkway
Sunnyvale, CA 94086

International Symposium on
Computer Performance Modeling,
Measurement, and Evaluation
North Holland
P O Box 103
1000 AC Amsterdam
The Netherlands

Performance Evaluation
(same as above)

Performance Evaluation Review
ACM/SIGMETRICS311 West 42nd Street
New York, NY 10036

Software Engineering
IEEE Computer Scoiety
10662 Los Vaqueros Circle
Los Alamitos, CA 90720

APPENDIX B
EXAMPLES OF PERFORMANCE MEASURES

The following performance measures were obtained from Svobodova [Ref. 12], and are grouped according to quanity of service, quality of work, component utilization, underlying factors, and workload characteristics. The measure is in bold type followed by the definition.

1. Quantity of work executable by a system
   a) **Throughput:** Amount of useful work peformed by the system per unit of time (job processing capability).
   b) **CPU productivity:** Percent of time spend in problem state (used as a measure of throughput).
   c) **Capacity:** Total work executable per time unit with a balanced workload.
   d) **System limits:** Number of terminals that can be supported or number of jobs than can be multiprogrammed without serious degradation of service.
   e) **Reserve capability:** Unused system capability.
2. Quality of Service
   a) **Turnaround time:** Elapsed time between submitting a job to the system and completion of ouput.
   b) **Response time:** Elapsed time between entering the last character of a request at a terminal and receiving the first character of the response. For one time slice tasks **Response time** is the elapsed time to complete for tasks that require less than one CPU time slice.
   c) **Gain factor:** Total active time needed to execute a job mix under multiprogramming/total active time needed to execute same mix sequentially.

d) **Elapsed Time Multiprogramming Factor (ETMF)**: Elapsed time to complete a task under multiprogramming/elapsed time to complete when this task is the only one active in the system.

e) **Internal Delay Factor (IDF)**: Active time needed to complete a task under multiprogramming/active time to complete when this task is the only one in the system.

f) **External Delay Factor (EDF)**: Task elapsed time/ task active time.

g) **Reliability**: Probability that the system functions correctly at any given time.

h) **Availability**: Percentage of time a user can get system services.

i) **Ease of use**: Time needed to prepare and debug programs for the system.

3. Component Utilization

a) **Resource utilization**: Percent of time a resource is in use.

b) **System utility**: Weighted sum of utilization of system resources.

c) **Component overlap (device gain)** : Percent of time operations of two or more hardware components are overlapped.

d) **Queue length**: Average queue length.

e) **Overhead**: Percent of CPU and channel time required by the operating system.

f) **Swap efficiency**: Frequency of occurence of individual instruction operation codes.

4. Underlying factors

a) **Computer power**: Number of operations per unit of time.

b) **Raw speed**: Actual speed of computer components.

c) **Reaction time:** The time that elapses from when an input arrives into the system until it recieves its first time slice.

d) **Internal response time:** Time between entering the last character on a terminal and receiving first CPU service.

e) **Program efficiency:** Amount of time spent executing individual portions of a program.

5. Workload Characteristics

a) **User interaction cycle:** Time between initiation of successive tasks from a single terminal.

b) **User interaction rate:** Frequency with which a single user request system services.

c) **Service time (compute time):** CPU time required by a single task (job).

d) **Service rate:** Rate at which requests are serviced by the CPU.

e) **Arrival rate:** Rate of task arrivals to the processor stage.

f) **Terminal time (user think-type time):** Time a task spends in the inactive state waiting for a user response.

g) **User intensity:** CPU time requested/user think-type time.

h) **I/O time:** Service time at an I/O processor.

i) **I/O request rate:** Amount of I/O service required by a single task.

j) **Active time:** CPU time per task core residence.

k) **Blocked time:** Time a task is incapable of receiving CPU service.

l) **Degree of multiprogramming:** Number of simultaneously active batch users or simultaneously active user terminals.

LIST OF REFERENCES

1. Morris, M. F. and Roth, P. F., Computer Performance Evaluation, Van Nostrand Reinhold Company, 1982.

2. Natioal Bureau of Stanards Special Publication 500-95, Proceedings of the Computer Performance Evaluation Users Group 18th Meeting "Development of A Standard Performance Management Strategy for the U.S. Navy", by S. B. Olson, October 1982

3. USAF Project Rand Report R-549-1-PR, Computer Performance Analysis: Framework and Initial Phases for a Performance Improvement Effort by Bell, T. E., Boehm, B. W. and Watson, R. A., November 1972

4. U.S. Department of Commerce/National Bureau of Standards FIPS PUB 49, Guidelines On Computer Performance Management: An Introduction May 1977

5. National Bureau of Standards Special Publication 500-95, Proceedings of the Computer Performance Evaluation Users Group 18th Meeting, "Information Systems Management", by J. J. Spinelli, October 1982

6. FEDSIM 2461 Eisenhower Avenue, Alexandria, Va. 22314 Phone (202) 325-0607 or Autovon 221-0607

7. Navy Regional Data Automation Center (NARDAC), Pensacola, Technical Support Department, Planning and Analysis Division, (Code312), Naval Air Station, Pensacola, Fl 32508

8. Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, DC 20234 Phone (301) 921-3485

9. System and Software Technology Division, Institute for Computer Sciences and Technology Report NBS-GCR-83-440 Capacity Planning: A State of the Art Survey by J. C. Kelly, July 1983

10. Kolence, Kenneth W., Introduction to Software Physics Institute for Software Engineering, Palo Alto, Ca. 1976

11. Jalics, P. J., "Improving Performance the Easy Way", Datamation pp 135-142, April 1977

12. Standford University Technical Report No. 72, Computer Performance Measurement and Evaluation Methods: Analysis and Applications by Svobodova, L. 1974

13. "Performance Monitors-Hardware or Software?", EDP Performance Review pp 1-8, May 1973

14. Bookmanm P.G.; Brotman, B.A; and Schmitt, K.L., "Use Measurement Engineering for Better System Performance", Computer Decisions pp 28-32, April 1972.

15. "Savings from Performance Monitoring", EDP Analyzer, pp 1-15, September 1972.

16. Sewald, M.D., Rauch, M.E., Rodiek, L., and Wertz, L., "A Pragmatic Approach to Systems Measurement", Computer Decisions pp 38-40, July 1971.

17. Carlson, G., "Fallacies of Intuition in Performance Tuning", EDP Performance Review pp 1-6, April 1976

18. Fisher, D., "Historical Fallacies", Toward a Logic of Historical Thought, Harper Torchbooks, 1970

19. Bailey, C., "Network Performance-Special Series On System Integration", Systems & Software pp 118-124, May 1983

20. Bogdanowicz, R., Crocker, M., Hsaio, D., Ryder, C., Stone, V. and Strawser, P., "Experiments in Benchmarking Relational Database Machines" a paper presented at the Third International Workshop on Database Machines, Munich, West Germany, September 1983.

21. NBS Special Publication 500-65 Proceedings of the Computer Performance Users Group 16th. Meeting, On the Simulation of Modeling Network DBMS, by Aitken, J.A and Hsu, H.T., pp 3-9, October 1980

22. Muchsel, R., "Performance Measurements--A Practical Example from a German University Computing Center", The Computer Journal, pp 188-192, Feburary 1982

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Post Graduate School Monterey, California 93943	2
3.	Curricular Officer Computer Technology Programs Code 37 Naval Post Graduate School Monterey, California 93943	1
4.	Gary K. Gray Rt 1 Box 838 Hollywood, Maryland 20636	3
5.	Dr. Alan A. Ross Code 52Rs Naval Postgraduate School Monterey, California 93943	4

206319

Thesis
G7162 Gray
c.1 A guide to Macro and Micro Computer Performance Evaluation.

206319

Thesis
G7162 Gray
c.1 A guide to Macro and Micro Computer Performance Evaluation.

AMERICAN BINDERY TOPEKA KANSAS

thesG7162
A guide to Macro and Micro Computer Perf
3 2768 002 13856 2
DUDLEY KNOX LIBRARY